Springer Series in Statistics

Thomas J. Santner
Brian J. Williams
William I. Notz

The Design and Analysis of Computer Experiments

Second Edition

Springer

Thomas J. Santner • Brian J. Williams
William I. Notz

The Design and Analysis of Computer Experiments

Second Edition

Springer

Thomas J. Santner
Department of Statistics
The Ohio State University
Columbus, OH, USA

Brian J. Williams
Statistical Sciences Group
Los Alamos National Laboratory
Los Alamos, NM, USA

William I. Notz
Department of Statistics
The Ohio State University
Columbus, OH, USA

ISSN 0172-7397 ISSN 2197-568X (electronic)
Springer Series in Statistics
ISBN 978-1-4939-8845-7 ISBN 978-1-4939-8847-1 (eBook)
https://doi.org/10.1007/978-1-4939-8847-1

Library of Congress Control Number: 2018953764

I should have read the whole book!!

To Gail, Aparna, and Claudia
for their encouragement and patience
during the writing of the whole book

Preface to the Second Edition

Computer experiments continue to increase in popularity as surrogates for and adjuncts to physical experiments. Since the publication of the first edition, there have been many methodological advances and software developments to implement these new methodologies. These advances have motivated our desire to update the book, and in this second edition, we have attempted to accomplish several things. First, we have included many of the advances in the design and analysis of computer experiments that have occurred since the first edition. Unfortunately, the subject continues to advance rapidly, and we will undoubtedly be "out-of-date" when this edition appears, but we hope this edition will be a useful resource by providing the material for understanding the foundations of the subject, by presenting methodology that we have found useful in our collaborations, and by pointing out the directions of other methodological advances in Chapter Notes sections. Second, we have tried to improve the presentation of the basic material on computer experiments and Gaussian processes with additional simulations and examples. The motivation for these changes have come from users and our own recognition of discussions needing improvement. Third, we have tried to write the majority of the text at an overview level that is accessible to readers with Masters-level training in Statistics while also discussing topics in sufficient detail to be useful for practitioners and researchers. While both objectives are not always simultaneously possible, we hope that all readers will be able to gain from the book.

To aid practitioners, the Chapter Notes provides lists of software that can be used to implement procedures discussed in this book. We make no claim of the completeness of these lists nor do our comments review its accuracy or ease of use. As of the publication date of this book, all links were tested and functioned properly but they may be updated at any time by those who maintain them. Indeed, many programs are in continuous development and web searches will provide up-to-date information about them as well as other packages that have been developed since the publication of this book.

The authors would like to thank colleagues who have either critiqued versions of this material or made use of it in courses. In particular, we thank Jeff Wu and

Roshan Joseph Vengazhiyil at Georgia Tech, David Steinberg at Tel Aviv University, Peter Marcy at Los Alamos National Laboratory, Peter Qian at the University of Wisconsin, and Jai-Hyun Byun at Gyeongsang National University. We give special thanks to our Ohio State colleague Angela Dean who carefully read parts of the manuscript and made numerous suggestions. Several Ohio State graduate students provided criticisms to drafts of the second edition as well as help with computations including Gang Han, Josh Svenson, Jeff Lehman, Fang-Fang Sun, Eric Lam, Arun Kumar, Soma Roy, Marian Frazier, Aaron Quan, Yulei Zhang, Erin Leatherman, Casey Davis, and Po-Hsu Allen Chen. Data contributors to the second edition include Shan Ba and William Myers at Procter & Gamble Company, and the Advanced Strategic Computing (ASC) program at Los Alamos National Laboratory. Co-workers/fellow researchers whose questions motivated some of the methodological advances described in the text include Derek Bingham at Simon Fraser University; Angela Dean, Chris Hans, and Matt Pratola at the Ohio State University; and Dave Higdon at Virginia Tech University.

We would like to thank those who have taken short courses that we have given on this material. Their queries have helped focus our presentation of some of this material. Finally, we would like to thank the staff at Springer Verlag for their encouragement and patience in the writing of this book. Any errors in the book are the responsibility of the authors; they would appreciate being informed.

Columbus, OH, USA — Thomas J. Santner
Santa Fe, NM, USA — Brian J. Williams
Columbus, OH, USA — William I. Notz
May 2018

Preface to the First Edition

Many physical processes are difficult or impossible to explore directly by conventional experimental methods. As computing power has increased, it has become possible to model some of these processes by sophisticated computer code. In such cases, the code can serve as a proxy for the physical process. As in a physical experiment, one can vary the inputs to the code and observe how the output is affected. Such studies are called computer experiments and are becoming increasingly popular surrogates for and adjuncts to physical experiments.

Much of the methodology for designing, modeling, and analyzing computer experiments can only be found in research papers. Our goal in writing this book is to make these methods accessible to a more general audience. To accomplish this, we have tried to keep the mathematics at a level that will be understandable to readers with Masters-level training in Statistics. This has been a challenging task. Gaussian processes are a popular way to model the output of computer experiments, but Gaussian process models are mathematically complex and likely to be unfamiliar to many readers. We provide an introduction to these models and present references for those who wish to study additional mathematical details of such processes. In other chapters, we relegate mathematical details to notes at the chapter end.

To make the book useful to practitioners, we provide software that can be used to fit the models we discuss in this book. Instructions for how to use this software can be found in Appendix C. Samples of the use of the software are interspersed throughout the book.

We would like to acknowledge several people for their contributions in bringing this book to completion. Don Bartel and members of the Biomechanics group at Cornell University and the Hospital for Special Surgery galvanized our initial interest in the area of computer experiments to investigate problems of prosthesis design. Several graduate students provided valuable criticism on early drafts and help with the computations including Jeff Lehman, Wei Zhao, Ofelia Marin, and Zhenhuan Cui. John Kimmel provided encouragement throughout the process; several anonymous reviewers gave us constructive suggestions. Antonia Orrant, Frank Ganz, and Frank McGuckin of Springer Verlag were the production and technical

assistance team that brought the work to print. Rob Tibshirani provided a critical Perl script that helped automate the compilation of the author index. The work of Brian J. Williams was supported by the Statistics Group at the RAND Corporation through core funding provided by James Thomson, President and Chief Executive Officer, and Michael Rich, Executive Vice President of RAND.

Columbus, OH, USA Thomas J. Santner
Santa Fe, NM, USA Brian J. Williams
Columbus, OH, USA William I. Notz

Contents

Chapter 1
Physical Experiments and Computer Experiments

1.1 Introduction

Experiments have long been used to study the relationship between a set of inputs to a physical system and the resulting output. Termed *physical experiments* in this text, there is a growing trend to replace or supplement the physical system used in such an experiment with a deterministic simulator. The resulting investigation is called a *computer simulator experiment* or more simply a *computer experiment*. The deterministic simulator, also called a *computer model*, implements in computer code a mathematical model relating input and output variables. A typical mathematical model might be a set of coupled partial differential equations, while common methods for solving the mathematical system include finite element (FE) and computational fluid dynamics (CFD) algorithms. This book describes methods for designing and analyzing research investigations that use computer simulator platforms, either alone or in combination with a physical experiment.

Historically, statistical science has devised numerous widely-used methodologies for designing *physical experiments* and for analyzing the resulting data to answer specific research questions. The process of designing a study to address such a question must decide which variables are to be observed and the role that each plays, e.g., as an explanatory variable or a response variable. The gold standard of data collection for establishing a cause and effect relationship uses a *prospective* design for the relevant physical experiment. *Agriculture* was one of the first sciences to apply widely, prospective designed experiments. Over time, alternative methods of conducting experiments have been developed to answer questions in subject matter areas having diverse constraints. For example, *controlled clinical trials* are used to compare medical therapies, while *case–control* studies were developed to answer questions in epidemiology.

The classical treatment of responses from a physical experiment considers them to be stochastic with a mean value that depends on a set of experimenter-selected *treatment variables* where knowledge of the effect of the treatment variables is the primary objective of the research study. However, many physical experiments also

T. J. Santner et al., *The Design and Analysis of Computer Experiments*, Springer Series in Statistics, https://doi.org/10.1007/978-1-4939-8847-1_1

consider the effects of other types of input variables. *Environmental* ("noise") *variables* describe *either* the operating conditions of an experimental subject/unit employing a given treatment *or* the conditions under which a manufacturing process is conducted. As an example of an environmental variable, consider an agricultural field trial of several test varieties that is conducted in multiple locations having different average growing season temperatures and rainfall. A *blocking factor* is a qualitative environmental variable that identifies identical groups of experimental material. Examples of blocking factors would be the use of gender or ethnicity to group subjects in a cancer clinical trial. *Confounding variables* are another type of input that can be present in a physical experiment. Confounding variables are unrecognized by the experimenter but actively affect the mean output of a physical system. Confounding variables have the potential to mask or exaggerate the effect of a treatment variable. For example, when an *active* confounding variable has values that are correlated with those of an *inert* treatment variable, the effect can be incorrectly attributed to the treatment variable.

Faced with these potential complications, statisticians have developed a variety of (design) strategies to increase the validity of treatment comparisons in physical experiments. One method is *randomization* which means the treatments are assigned to experimental units at random and are applied in a random order. A randomized experimental design has a smaller chance that the effect of a treatment variable will be misinterpreted due to the presence of a highly correlated and active confounding variable. Another technique to increase experimental validity is *blocking* which means that the experimental units are grouped to be as similar as possible. An experimental design that allocates treatments in a balanced way within blocks allows for valid within-block treatment comparisons. Using adequate *replication* is a third technique for increasing the validity of an experiment. Roughly, adequate replication means that an experiment is run on a sufficiently large scale to prevent unavoidable "measurement" variation in the output from obscuring treatment differences.

As for physical experiments, the inputs to a simulator used in a computer experiment can be classified according to the role they play (treatment, environmental, or other). In the computer experiment literature, treatment variables are most often called *control* variables. Computer simulators often include a class of variables not found in physical systems which are called *model* or *calibration variables* in this book. Examples of model variables are unknown rates of change, the friction between materials, and material properties. Model variables are often only partially known from previous research with uncertainty specified in a probability distribution reflecting expert opinion. Where it is useful to differentiate the types of variables to a simulator, this book will use the notation $\boldsymbol{x}_c$ to denote control variables, $\boldsymbol{x}_e$ to denote environmental variables, and $\boldsymbol{x}_m$ to denote model variables.

Computer simulator experiments have additional characteristics not found in physical experiments. The first is that they yield *deterministic output*, at least up to "numerical noise." The simulator produces the same answer if run twice using the same set of inputs. For this reason, blocking the runs into groups that represent "more nearly similar" experimental units is irrelevant. Indeed, none of the traditional principles of blocking, randomization, and replication used when conducting a physical experiment is relevant when conducting a computer simulator experiment.

Second, many computer experiments can be *time-consuming* to conduct. It would not be unusual for a detailed finite element model to run for 12 or more hours; the same is true of computational fluid dynamics models.

Third, the output of a computer simulator, being based on a mathematical model of the input/output relationship, *can exhibit* bias because either the mathematical model omits elements of the physics or biology that characterize this relationship or the numerical method used to implement the mathematical model lacks accuracy for some subdomain of the input space. One purpose of calibrating a simulator to data from an appropriate physical experiment is to estimate the simulator bias.

A fourth feature of many computer experiments is that the *number of input variables can be quite large*—20 or 30 inputs for some applications and hundreds of inputs for others. One reason for the large number of inputs to some codes is that they contain control, environmental, and model variables. As a simple example, consider the biomechanical design of a knee implant to minimize the maximum *strain* that occurs at the bone $\times$ prosthesis boundary of a knee implant. Multiple variables are required to describe the prosthesis geometry and its material properties (the control inputs for the biomechanical engineer). Among other subject-specific quantities, the strain depends on the magnitude of the load (an environmental input) and the friction between the prosthesis and the bone (a model input).

Using experiments based on simulator platforms alone or in conjunction with physical systems can have substantial advantages. For example, the process of calibration using simulator and physical system data, yields an estimated bias of the simulator. Examining the bias can suggest where improvements can be made to the simulator. Another advantage is that, in some applications, computer simulator experimentation can be feasible when physical experimentation is either not possible or is very limited. For example, the number of input variables may be too large to consider performing a physical experiment, there may be ethical reasons why a physical experiment cannot be run, or it may simply be economically prohibitive to run an experiment on the scale required to gather sufficient information to answer a particular research question. In such cases, depending on the amount of fundamental physics or biology implemented in the simulator, there are a growing number of scientific and technological investigations that have been conducted using simulator platforms.

The remainder of this chapter will provide several motivating examples that illustrate key features of computer experiments and show their broad application. Section 1.4 will provide an overview of the remainder of the book.

1.2 Examples of Computer Simulator Models

This section presents several examples where computer simulator models are used. The examples are meant to illustrate the elements of computer simulator experiments that have been sketched in Sect. 1.1. They also show the *breadth* of application of such models. The details of the mathematical models implemented by the computer code will not be given, but references to the subject matter literature will be provided. The first examples have univariate output, then several examples illustrate applications with multivariate output, and the final two examples describe settings that produce functional output.

Example 1.1 (Temporal Evolution of Fires in Enclosed Areas). Deterministic computer models are used in many areas of fire protection design including egress (exit) analysis. This example describes one of the early "zone" computer models that is used to predict the fire conditions in an enclosed room. Cooper (1980) and Cooper and Stroup (1985) provided a mathematical model and its implementation in FORTRAN for describing the evolution of a fire in a single room with closed doors and windows that contains an object that has been ignited at a point below the ceiling (see Sahama and Diamond (2008) for a simplified description of the mathematical model). The room is assumed to contain a small leak at floor level to prevent the pressure from increasing in the room. The fire releases both energy and hot combustion by-products. The rate at which energy and the by-products are released is allowed to change with time. The by-products form a plume which rises toward the ceiling. As the plume rises, it draws in cool air, which decreases the plume's temperature and increases its volume flow rate. When the plume reaches the ceiling, it spreads out and forms a hot gas layer whose lower boundary descends with time. There is a relatively sharp interface between the hot upper layer and the air in the lower part of the room which, in this model, is considered to be at air temperature. The only interchange between the air in the lower part of the room and the hot upper layer is through the plume. The model used by these programs can therefore be thought of as a two-zone model.

The Cooper and Stroup (1985) code is called ASET (Available Safe Egress Time). Walton (1985) implemented their model in BASIC, calling his computer code ASET-B; he intended his program to be used in the first generation of personal computers available at that time of its development. ASET-B is a compact, easy-to-run program that solves the same differential equations as does ASET but using a simpler numerical technique.

The *inputs* to ASET-B are

- the room ceiling height and the room floor area,
- the height of the burning object (fire source) above the floor,
- a heat loss fraction for the room (e.g., which depends on the insulation in the room),
- a material-specific heat release rate, and
- the maximum time of the simulation run.

The program *outputs* are the *temperature* of the hot smoke layer and its *distance* above the fire source as a function of time.

Since these early efforts, computer codes have been written to model the evolution of wildfires as well as fires in confined spaces. As typical examples of this work, see Lynn (1997), Cooper (1997) and Janssens (2000), and the references therein.

The publications of the Building and Fire Research Laboratory of NIST contain a wealth of material about mathematical modeling of fires (see http://fire.nist.gov/bfrlpubs). The review article by Berk et al. (2002) describes statistical approaches for the evaluation of computer models for wildfires. Sahama and Diamond (2001) give a case study using the statistical methods introduced in Chap. 3 to analyze a set of 50 observations computed from the ASET-B model.

To provide a sense of the effect of each of these variables on the evolution of the fire, the simulations run below *fix* the heat release rate to correspond to fire material that constitutes a "semi-universal" fire. This heat release profile corresponds to a fire in a "fuel package consisting of a polyurethane mattress with sheets and fuels similar to wood cribs and polyurethane on pallets and commodities in paper cartons stacked on pallets" (Birk (1997)). Then the remaining four factors were varied using a "Sobol´ design" (Sobol´ designs are described in Sect. 5.6.1). The response for this example is the time, to the nearest second, for the fire to reach 5 ft above the burning fuel package.

Scatterplots were constructed of each input versus the time required by the hot smoke layer to reach 5 ft above the fire source. Only room area showed strong visual associations with the output; Fig. 1.1 shows this scatterplot (see Fig. 3.3 for scatterplots of all four inputs versus the time to reach 5 ft above the fire source). This makes intuitive sense because more by-product is required to fill the top of a large room, and, hence, longer times are required until this layer reaches a point 5 ft above the fire source. The data from this example will be used later to illustrate several analysis methods. ♦

Example 1.2 (Formation of Pockets in Sheet Metal). Montgomery and Truss (2001) discussed a computer model that determines the failure depth of symmetric rectangular pockets that are punched in automobile-grade steel sheets; the failure depth is the depth at which the sheet metal tears. Sheet metal, suitably formed in this manner, is used to fabricate many parts of automobiles. This application is but one of many examples of computer models used in the automotive industry.

Rectangular pockets are formed in sheet metal by pressing the metal sheet with a punch (the target shape) into a conforming die. There are *six inputs* to the Montgomery and Truss (2001) code, all of which are engineering design variables. These variables can either be thought of as characteristics of the punch/die machine tool used to produce the pockets or, in most cases, as characteristics of the resulting pockets.

Five of the variables can be easily visualized in terms of the pocket geometry. In a top view of the pocket, Fig. 1.2 illustrates the *length* l and the *width* w of the rectangular pocket. In a side view of the pocket, Fig. 1.3 shows the *fillet radius* f which is the radius of the circular path that the metal follows as it curves from the flat upper metal surface to the straight portion of the pocket wall; the length of this region (measured horizontally on the pocket floor) is denoted R in both the side and top views of the pocket. The same fillet radius is followed as the straight portion of the pocket wall curves in a circular manner to the pocket floor; the length of this region (measured horizontally on the pocket floor) is denoted r in both views. Viewed from the top in Fig. 1.2 and from the side in Fig. 1.3, the *clearance* c is the

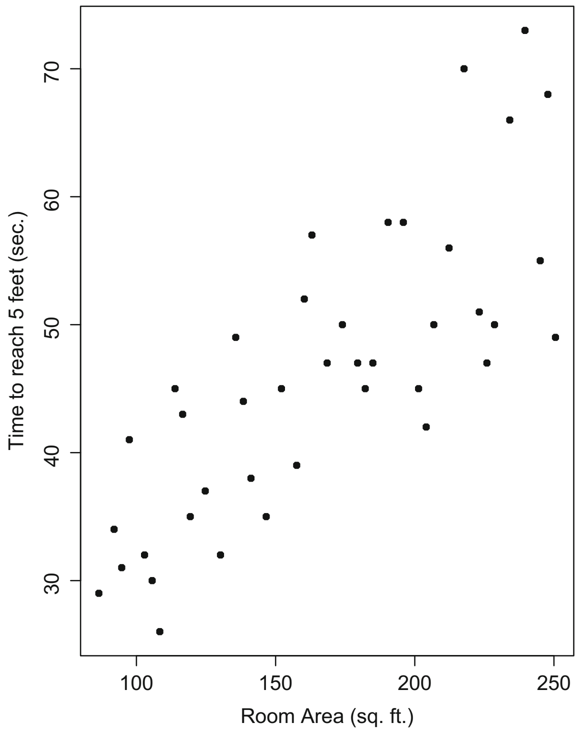


Fig. 1.1 Scatterplot of room area versus the time for the hot smoke layer to reach 5 ft above the fire source

horizontal distance during which the side wall of the rectangular pocket descends linearly. In terms of the punch/die machine tool, the clearance is the distance between the punch and the die when the punch is moved to its maximum depth within the die. The *punch plan view radius p* is described in Fig. 1.2. Shown in Fig. 1.3, the *lock bead distance d* is measured at the pocket edge on the top metal surface; the machine tool fixes the sheet metal at the top of the pocket so that it cannot stretch beyond distance *d* from the pocket edge.

To provide a sense of the (marginal) effect of each of these variables on the failure depth, the failure depth was plotted versus each of the six explanatory inputs for the set of 234 runs analyzed by Montgomery and Truss (2001). Two of these scatterplots are shown in Fig. 1.4; they are representative of the six marginal scatterplots. Five variables are only weakly related to failure depth, and the panel in Fig. 1.4 showing failure depth versus fillet radius is typical of these cases. One variable, clearance, shows a strong relationship with failure depth. ♦

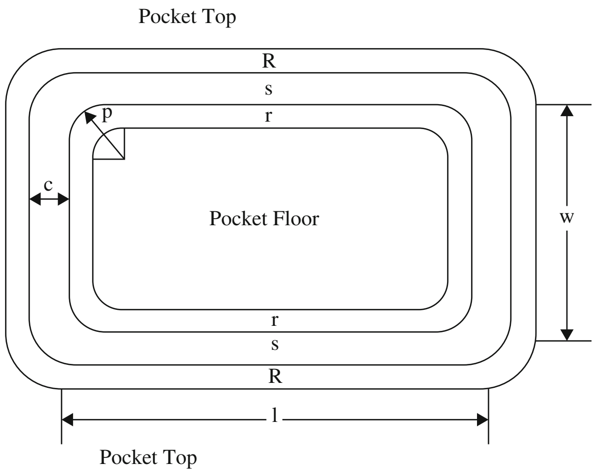


Fig. 1.2 Top view of the pocket formed by a punch and die operation. The floor of the pocket is the innermost (rounded-corner) rectangle. The letters *R*, *s*, and *r* correspond to the similarly labeled horizontal distances in the Fig. 1.3 side view

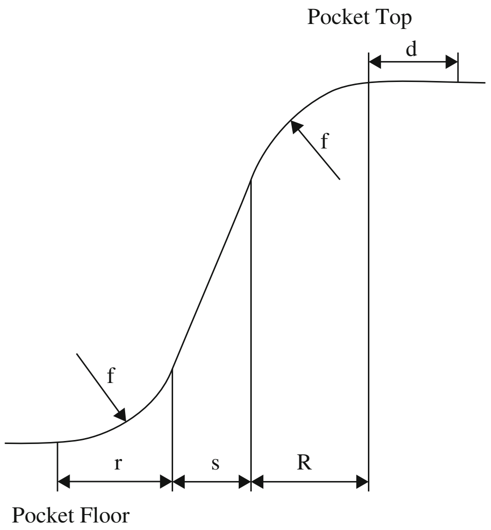


Fig. 1.3 Side view of part of a symmetric pocket formed by a punch and die operation. The angled side wall is created by the same fillet radius at the top by the die and at the bottom by the edge of the punch

Example 1.3 (Water Flow Through a Borehole). In their studies of uncertainty analysis, Harper and Gupta (1983) and Worley (1987) used a simple model of the flow of water through a borehole that is drilled from the ground surface through two aquifers. This model is based on the assumptions: no groundwater gradient, steady state flow from the upper aquifer into the borehole and from the borehole into the

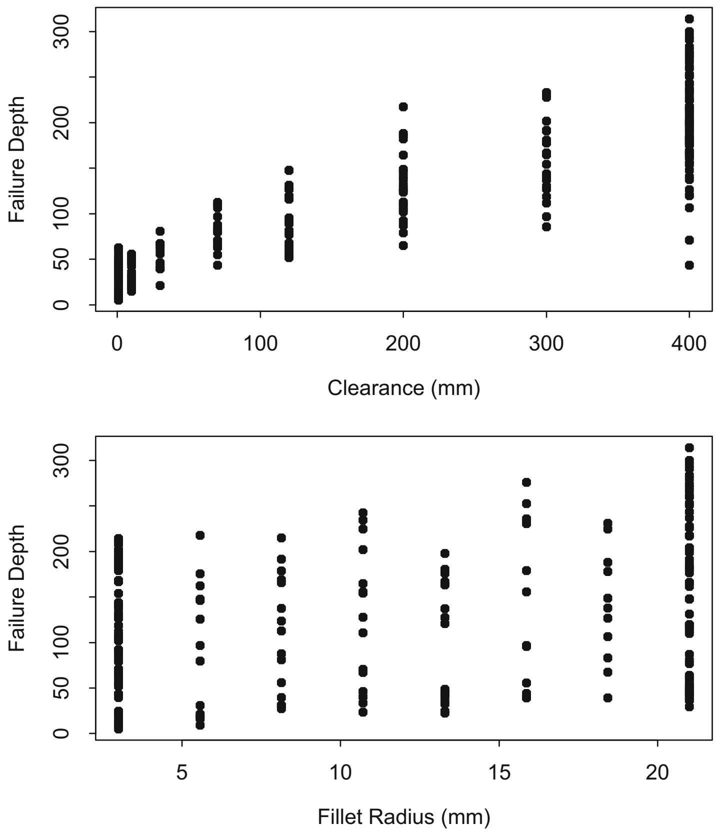


Fig. 1.4 Top panel: scatterplot of failure depth (millimeters) versus clearance for 234 runs of the computer code described in Example 1.2; bottom panel: failure depth versus fillet radius for the same data

lower aquifer, laminar fluid flow in parallel layers with no disruption between layers, and isothermal (constant temperature) flow through the borehole. Let $y_B(\boldsymbol{x})$ be the *flow rate* through the borehole in m^3/year. The Harper and Gupta (1983) and Worley (1987) model is

$$y_B(\boldsymbol{x}) = \frac{2\pi T_u(H_u - H_l)}{\ell n(r/r_w)\left[1 + \frac{2LT_u}{\ell n(r/r_w)\, r_w^2 K_w} + \frac{T_u}{T_l}\right]}$$

where the $\boldsymbol{x}$ inputs are labeled

r_w = radius of the borehole (m), $r_w \in [0.05, 0.15]$

T_u = transmissivity of the upper aquifer (m^2/year), $T_u \in [63{,}070, 115{,}600]$

$$\begin{aligned}
r &= \text{radius of influence (m)}, \ r \in [100, 50{,}000] \\
H_u &= \text{potentiometric head of the upper aquifer (m)}, \ H_u \in [990, 1110] \\
T_l &= \text{transmissivity of the lower aquifer (m}^2\text{/year)}, \ T_l \in [63.1, 116] \\
H_l &= \text{potentiometric head of the lower aquifer (m)}, \ H_l \in [700, 820] \\
L &= \text{length of the borehole (m)}, \ L \in [1120, 1680] \\
K_w &= \text{hydraulic conductivity of the borehole (m}^2\text{/year)}, \ K_w \in [9855, 12{,}045] \, .
\end{aligned}$$

Here $y_B(\boldsymbol{x})$ is not a computationally intensive computer model but rather is an analytic function with eight inputs. The ranges of each input are quite different. Because it is rapidly computable, $y_B(\boldsymbol{x})$ is useful for demonstrating some of the methodology discussed later in the book. For example, it allows us to quickly assess the accuracy of predictions at test sites by means of direct $y_B(\boldsymbol{x})$ comparisons. It is also useful for comparing new methodology to existing procedures. ♦

Example 1.4 (Injection Molding). Injection molding (IM) is one of the most important manufacturing methods for mass-producing precision plastic components that might be used in automobiles and appliances. This example describes part of a larger study to determine the *manufacturing conditions*, the "process control variables," to best achieve target dimensions for a suite of test parts fabricated by IM. Roughly, the steps of the IM process to fabricate plastic components are to first melt plastic pellets, then to inject the molten plastic material into a mold, and lastly to allow the molded material to cool for a specified length of time.

The goal is to find molding process conditions that minimize the *relative shrinkage* of the length, width, and thickness of a specified segment of a test sample. The relative shrinkage is defined as the absolute value of the difference between the target and manufactured measurement divided by the target measurement. To simplify the description of this example and the subsequent calibration in Chap. 8, *only the width* of a particular test sample will be considered.

Given adequate resources, a series of experiments might be conducted during *routine production* to minimize the relative shrinkage of fabricated test pieces. Unfortunately, it is often not feasible to run experiments that change the process control variables during standard production, and thus this approach was not used here. Instead, the physical system data used in this example were collected at an injection molding laboratory located on The Ohio State University campus. The control variables used in the physical experimentation, and their notation, were: (1) the melt temperature (T_{melt}), (2) the packing time (t_{pack}), (3) the packing pressure (P_{pack}), and (4) the cooling time (t_{cool}). The units of measurement of each control variable and its range are listed in Table 1.1. A total of 19 distinct settings of $(T_{melt}, t_{pack}, P_{pack}, t_{cool})$ were used in a physical experiment, shown in Table 1.2 along with the observed data. The boxplot in the right portion of Fig. 1.5 shows the distribution of the relative shrinkages of the widths of test pieces manufactured using the 19 process control variable settings.

Variable	Symbol	Type	Low	High
Melting temperature (°C)	T_{melt}	Control	180	220
Packing time (s)	t_{pack}	Control	10	30
Packing pressure (MPa)	P_{pack}	Control	30	44
Cooling time (s)	t_{cool}	Control	25	50
HTC flow (W/m^2 K)		Calibration	1200	1800
HTC pack (W/m^2 K)		Calibration	20,000	30,000
HTC open (W/m^2 K)		Calibration	2000	3000

Table 1.1 Control and calibration input variables for the IM process of Example 1.4

T_{melt}	t_{pack}	P_{pack}	t_{cool}	$y^p(\boldsymbol{x}^p)$
184	14	42.67	50	0.0057
184	14	32.14	36	0.0056
184	14	42.67	36	0.0057
184	28	42.67	36	0.0051
184	21	37.41	43	0.0048
184	14	32.14	50	0.0067
184	28	32.14	50	0.0061
184	28	42.67	50	0.0049
184	28	32.14	36	0.0066
216	14	42.67	36	0.0046
216	28	42.67	36	0.0047
216	14	42.67	50	0.0041
216	14	32.14	36	0.0054
216	21	37.41	43	0.0046
216	28	42.67	50	0.0038
216	28	32.14	36	0.0048
216	14	32.14	50	0.0056
216	28	32.14	50	0.0052
200	21	37.41	43	0.0054

Table 1.2 The 19 distinct combinations of control factor settings used in the physical experiment of Example 1.4 and the corresponding relative shrinkage of the width of the test part

Alternative to experimenting with an existing manufacturing process or conducting an off-line study, many molders experiment using a computer simulator of the molding process (see Zhou (2013)). The simulator employed in this example is Moldex3D, a commercial package that implements a mathematical model which treats the plastic melt as a generalized Newtonian fluid (Villarreal-Marroquín et al. (2017), and the references therein provide more information about the mathematical model; see also http://www.moldex3d.com). The inputs to Moldex3D include the four process control variables together with three variables used to calibrate the simulator output to the physical output; the calibration variables are the heat transfer coefficients (HTC) of the mold during flow, packing, and cooling. In the manufacturing setting, the HTCs cannot be determined with 100% accuracy, but expert opinion about their possible values is listed as the ranges in Table 1.1. This example uses combined physical system and simulator data to form calibrated predictors of the mean output from the physical system at arbitrary control inputs (see Sect. 8.3).

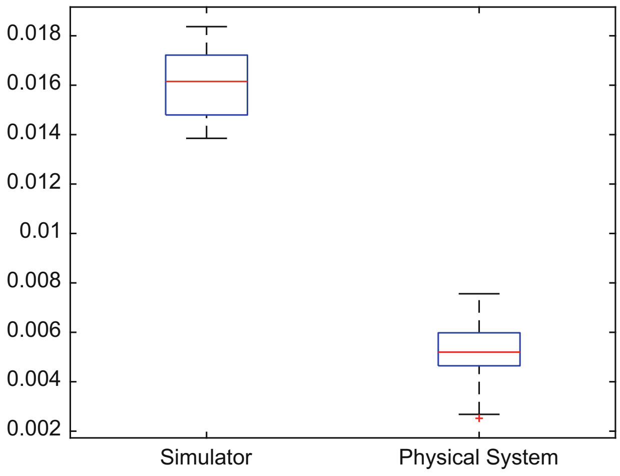


Fig. 1.5 Boxplots of the relative shrinkages of the widths of test pieces from the 35 simulator run outputs (left) and 19 physical system observations (right) of Example 1.4

A 35 run space-filling maximin Latin hypercube design was selected at which to run the simulator (see Sect. 5.4 for a description of maximin Latin hypercube designs). The space-filling character of these seven inputs can be seen in the scatterplot matrix of Fig. 1.6 which shows all 21 $\left(= \binom{7}{2}\right)$ 2-D projections of these seven variables. The left side of Fig. 1.5 is a boxplot of the computed relative shrinkages of the width of the test piece for the 35 runs. The fact that the 35 outputs for the simulator are systematically greater than those from the physical experiment is visual evidence of a bias in the simulator code that appears to require more adjustment than mere manipulation of the HTCs. ♦

Example 1.5 (Design of Prosthetic Devices). Biomechanical engineers have used computer simulators to understand the performance of total joint replacements for decades. Their simulator models supplement clinical studies and limited physical experiments. Simulator models are used for design comparisons and optimization.

Two important difficulties that limit the use of many simulator models are their long running times and their difficult validation. For example, even using a fast workstation, a single run of many simulator codes can require hours or even days. The validation of a simulator is the process of ensuring that the theoretical model which is the basis for the simulator correctly describes joint performance.

As a simple example of a biomechanics simulator, Chang (1998) and Chang et al. (2001) study the design of a reduced midstem "bullet" tip prosthesis. As illustrated in Fig. 1.7, the total length of the stem was fixed at 100 mm, and its maximal diameter at the tip was 16 mm. Figure 1.7 also illustrates the biomechanical design variables which were the length $b \in [25\text{ mm}, 75\text{ mm}]$ of the maximum diameter portion of the stem and diameter of the reduced midshaft $d \in [7\text{ mm}, 13\text{ mm}]$.

The finite element code of Chang (1998) considers five inputs, $\boldsymbol{x}$. Two are the design variables (b, d) described in the previous paragraph. The remaining three

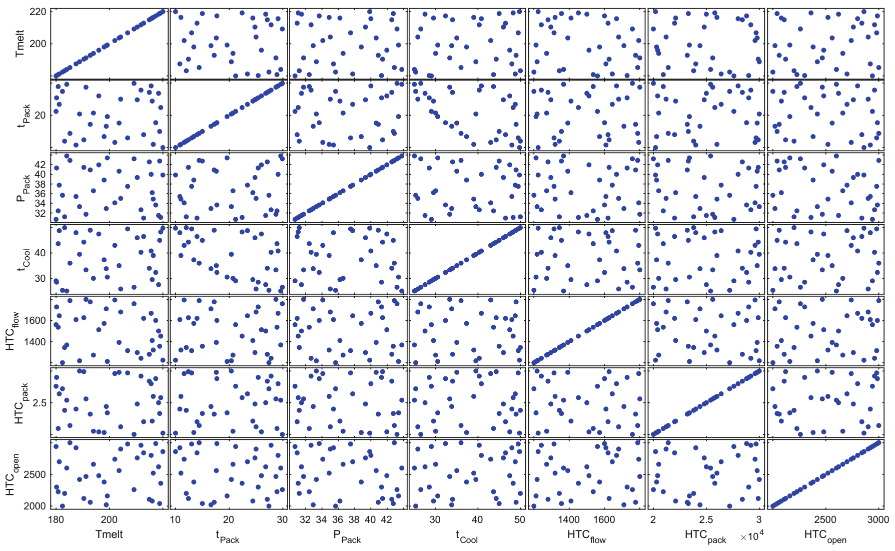


Fig. 1.6 Scatterplot matrix of all 2-D projections of the control and calibration inputs for the 35 simulator runs of Example 1.4

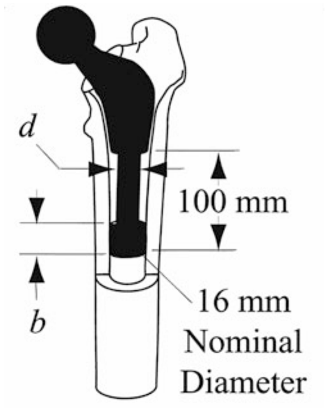


Fig. 1.7 Biomechanical engineering design variables b and d for variation of a Ranawat–Burstein hip prostheses having a fixed stem length of 100 mm and a maximum diameter of 16 mm

inputs are patient-specific variables which allowed for differences in the simulated performance of each (b, d) design over a population of patients. The patient-specific variables were

- Θ, the joint force angle (the load itself was fixed),
- E, the elastic modulus of the (cancellous) bone surrounding the implant, and
- F, the bone-implant interface friction.

Figure 1.8 illustrates the (Θ, E) inputs. The range of values of the three patient inputs in human populations were determined from the orthopedic literature. For example, the joint force angle was determined in telemetric hip force studies (Kotzar et al. (1995)).

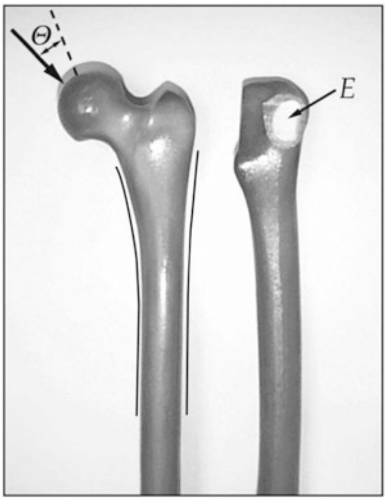


Fig. 1.8 Two of the patient-specific inputs to the simulator of Chang et al. (2001): Θ, the joint force angle, and E, the elastic modulus of the bone surrounding the implant

The simulator produced two outputs for each design × patient condition, $\boldsymbol{x} = (b, d, \Theta, E, F)$. One output is *femoral stress shielding*, denoted by $y_s(\boldsymbol{x})$, and a second is *implant toggling*, denoted by $y_t(\boldsymbol{x})$. The femoral stress shielding measures the amount of the load that is deflected from the top of the femur near the neck of the prosthesis. While maximizing shielding might be thought to be good, the opposite is true. Because bone is a living organism that reacts to its environment, shielding the femur from stress causes it to lose bone stock, and hence the femur weakens over time. Implant toggling measures the flexing of the implant in the coronal plane of the body (a vertical plane that divides the body into its anterior and posterior sections). Implant toggling should also be minimized because excessive flexing in the coronal plane can cause implant loosening.

The simulator outputs $y_s(\boldsymbol{x})$ and $y_t(\boldsymbol{x})$ are competing objectives. Flexible prostheses minimize stress shielding but permit the prosthesis toggling and thus increase the chance of loosening. A prostheses that is too stiff will not toggle in the coronal plane but will shield the bone that is near the neck of the prosthesis causing this section of the bone to be gradually resorbed by the body and weakened. The modified bone structure can be more fragile than that of a normal person and can fracture more easily under the stress of, say, a fall. Combining multiple objectives is typically problematic, and Sect. 6.3 describes the Pareto approach for doing so. ♦

Example 1.6 (Stability of Acetabular Cups Under Cyclic Loading). This example, taken from Ong et al. (2008), is drawn from the biomechanical engineering literature. It illustrates a simulator with multivariate output that has both control and environmental inputs. Ong et al. (2008) study the design of the so-called *acetabular cup* which is a component of a prosthetic total hip replacement. The acetabular cup is located in the hip socket ("acetabulum") of the pelvis. The head of the femoral component of the prosthesis conforms to the cup and rotates within it (see the Xray in Fig. 1.9).

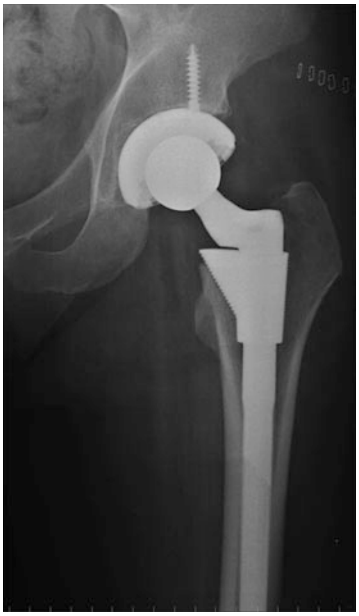


Fig. 1.9 MRI of a total hip replacement that shows the femoral component and acetabular cup. The acetabular cup is located in the pelvis (Credit: NIADDK, 9AO4 (Connie Raab-contact); NIH)

One cause of failure of the hip prosthesis is the loosening of the acetabular cup over time. Cup loosening is caused by the buildup of polyethylene debris from the interior of the cup which is, in turn, the result of friction between the femoral head and the cup interior. Cup designs that have relatively little movement between the pelvic bone and back of the cup are less likely to "suction" debris into this space and are considered desirable.

The 3-D finite element simulator used to study alternative cup designs had 12 inputs that can be grouped as

- two engineering design ("control") inputs that described deviations of the cup from sphericity in two directions,
- two variables that captured patient loading,
- one input that described model assumptions (a friction coefficient),

- seven inputs that measured surgical skill in placement of the cup and the reaming of the hemispherical bed in the pelvic bone for the acetabular cup.

The cartoon in Fig. 1.10 illustrates the engineering design, cup placement, and loading variables. In more detail, Table 1.3 lists all the inputs, the domain of possible values for the engineering design inputs, and gives the distributions used in Ong et al. (2008) to describe the relative frequency of occurrence of the loading inputs, the level of surgical skill, and the subject expert beliefs about the value of friction in this application.

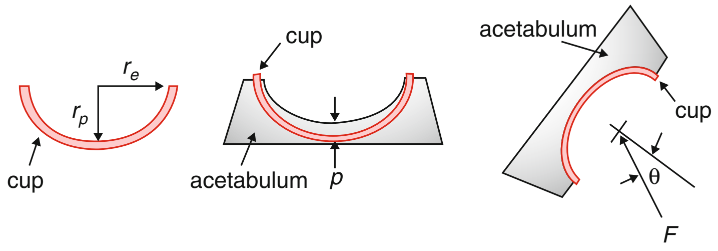


Fig. 1.10 Left panel, schematic of two engineering design variables, the cup equatorial diameter ($2 \times r_\theta$) and the cup eccentricity ($2 \times r_\theta - 2 \times r_\rho$); center panel, one of the surgical inputs, p the depth of cup insertion; right panel, two inputs, F and θ, describing patient loading

The simulator calculated four (related) outputs that are proxy measures of the cup stability under loading (termed *total potential ingrowth area*, *change in gap volume*,

Input	Parameter	Input range or distribution
	Bioengineering design variables	
1	Cup equatorial diam. (mm)	{56, 57, 58, 59}
2	Cup eccentricity (mm)	{0, 1, 2}
	Patient-specific environmental variables	
3	Gait load magnitude (BW)	Chi-square truncated
4	Gait load polar direction (deg)	Truncated Normal distribution
	Model input	
5	Density/modulus relative weight	Trangular(0, 1)
	Surgical environmental variables	
6	Cup penetration–insertion (mm)	Truncated Normal distribution
7	Nominal reaming deviation at equator (mm)	Truncated Normal distribution
8	Nominal reaming deviation at pole (mm)	Truncated Normal distribution
9	Frequency of undulations (cross-section)	$U(4, 9)$
10	Frequency of undulations (transverse section)	DU{4, 5, 6, 7}
11	Peak amplitude of undulations (mm)	$U(0.85, 1.15)$
12	Rate of undulation decay	$U(0.85, 1.15)$

Table 1.3 Biomechanical design inputs (#1 & #2) and their input domains; environmental variables (#3-#12) and a brief description of the distribution of the occurrence of each in the population studied in Ong et al. (2008). Here $U(a, b)$ denotes the uniform distribution over the interval (a, b); $DU\{a_1, \ldots, a_d\}$ denotes the discrete uniform distribution over the values $\{a_1, \ldots, a_d\}$; and $Tr(a, b)$ denotes the triangular distribution centered at $(a + b)/2$

gap volume, and *cup relative motion* in Ong et al. (2008)). Among other issues, the sensitivity of the outputs to each input will be determined in later examples. ♦

Example 1.7 (Flyer Plate Experiments). In a flyer plate experiment, a plane shock wave is forced through a stationary target sample of a material. Flyer plate experiments are conducted to learn about the behavior of materials in high strain-rate environments. In the data analyzed in later sections, the target material is the element tantalum for which data from a single physical experiment was available.

Figure 1.11 is a cartoon that provides more detail about the test apparatus used in a flyer plate experiment. A shock wave is caused by an (aluminum) impactor plate that is forced into a target plate by the detonation of a high explosive charge in a closed test chamber. The output of the experiment is the *velocity of the free surface of the target plate* measured as a function of time. The free surface of the target plate is the face opposite the surface that is hit by the impactor plate. In Fig. 1.11 the free surface is located on the bottom outside of the test chamber. A set of "shorting-pin detectors" measures the velocity of the face as a function of time.

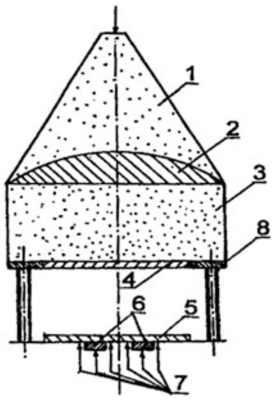


Fig. 1.11 Diagram of flyer plate experiment with an accelerated aluminum impactor: (1) lens-shaped high-explosive charge; (2) correcting lens; (3) main charge; (4) impactor plate; (5) shield from a standard material; (6) target sample; (7) shorting-pin detectors; (8) peripheral steel ring (adapted from Trunin (1998))

The computer simulator of a flyer plate experiment that was used throughout this example was based on a two-dimensional hydrodynamic mathematical model (with code developed at Los Alamos National Laboratory). The code incorporated physics models that required specification of model parameters that would be unknown in a corresponding physical experiment. Broadly, there are three physics models that were implemented in the simulator: an *equation of state* (EOS) model, a *material strength* model, and a *material damage* model. The EOS model states the mathematical relationship between two or more states associated with a given material, such as the material's temperature, volume, or internal energy. The material strength model specifies the behavior of a solid material under the action of external loading; the so-called "Preston–Tonks–Wallace (PTW)" model (Preston et al. (2003)) is used by the code. The material damage model describes the amount of spallation on the free surface; spallation is the process by which fragments of material (the "spall") are discharged from the free surface of the target plate. Spall can be due to impact or stress. The spall acts as a secondary projectile with velocities that can be a substantial fraction of the speed of the stress wave impacting the material.

Each code run (and the physical experiment) produced functional data—the velocity profile of the free surface at 136 time points. A generic velocity profile of the free surface of a target material is shown in the left panel of Fig. 1.12. The velocity profile from a set of simulator runs for tantalum is shown in the right panel of Fig. 1.12.

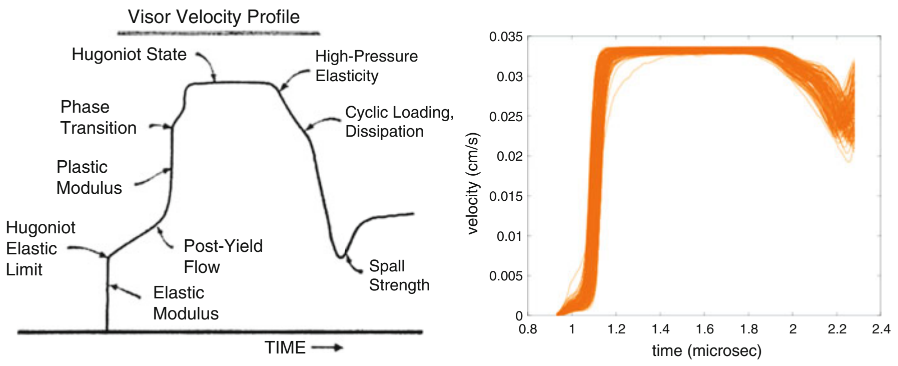


Fig. 1.12 Left panel, nomenclature for the components of a free-surface velocity profile; right panel, free-surface velocity profiles from 128 runs of a simulator for a flyer plate experiment using tantalum

The simulator had ten inputs; Table 1.4 lists the inputs and their ranges. The following sections analyze the results from 128 code runs. Figure 1.13 shows that the input settings fill each 1-D and 2-D projection of the input space. The inputs were selected according to an orthogonal array-based Latin hypercube design—see Sect. 5.3.

In Chap. 8 on calibration, the data from the simulator runs and the physical experiment are used to infer tantalum's material properties, i.e., the parameters of its equations of state, its material strength, and its material damage. ♦

Example 1.8 (Location $\times$ Time Profile of the Concentration of a Chemical from a Bi-location Spill). Bliznyuk et al. (2008) present a pseudo-diffusion model for the location $\times$ time evolution of the concentration of a chemical caused by a pair of pollutant spills. The model considers a long narrow channel and allows the spills to occur at different locations and times. Let (s, t) denote a generic location and time where the concentration is to be measured.

The mathematical model assumes that diffusion is the only transport modality. The diffusion rate in the channel is denoted D. For simplicity their model assumed

Input	Symbol	Description	Domain	
			Min	Max
1	ϵ	Perturbation of EOS table from nominal	−5%	+5%
2	θ_0	Initial strain hardening rate	2.78×10^{-5}	0.0336
3	κ	Material constant in thermal activation energy term (relates to the temperature dependence)	0.438	1.11
4	γ	Material constant in thermal activation energy term (relates to the strain-rate dependence)	6.96×10^{-8}	6.76×10^{-4}
5	y_0	Maximum yield stress (at 0 K)	0.00686	0.0126
6	y_∞	Minimum yield stress (at ~ melting temp)	7.17×10^{-4}	0.00192
7	s_0	Maximum saturation stress (at 0 K)	0.0126	0.0564
8	s_∞	Minimum saturation stress (at ~ melting temp)	0.00192	0.00616
9	$P_{\min}$	Spall strength	−0.055	−0.045
10	v_s	Flyer plate impact velocity	329.5	338.5

Table 1.4 Inputs to flyer plate simulator: input 1 specifies the equation of state model; inputs 2–8 specify the material strength model; input 9 specifies the material damage model

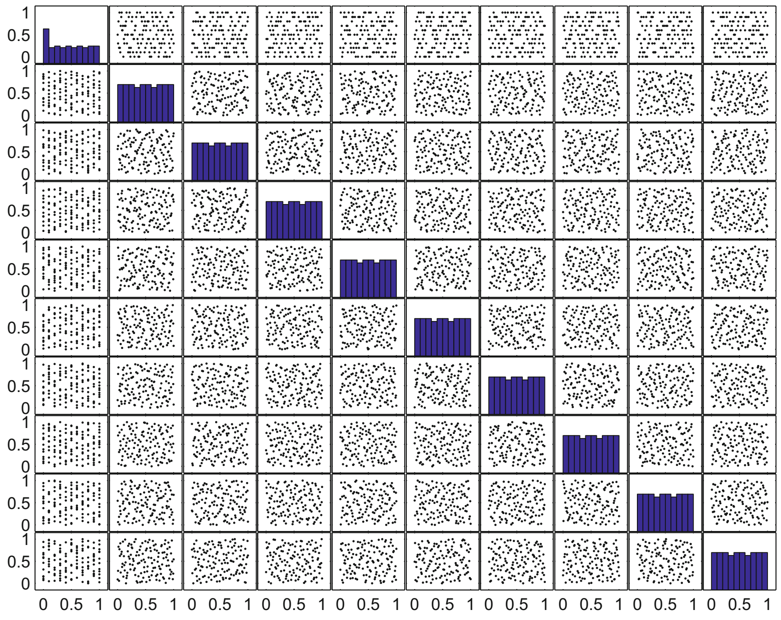


Fig. 1.13 Scatterplot of all pairs of the normalized inputs used in the flyer plate simulator runs

both spills contained the *same* mass M of the pollutant. Denoting the location and time of the first spill by $(s, t) = (0, 0)$, the concentration profile depends on the location and time of the 2nd spill which are denoted by $(s, t) = (L, T)$. The ranges of the inputs (M, D, L, T) are as follows:

$$M = \text{common mass of the pollutant spilled, } M \in [7, 13]$$

$$D = \text{diffusion rate in the channel, } D \in [0.02, 0.12]$$

$$L = \text{location of the second spill, } L \in [0.01, 3.0]$$
$$T = \text{time of the second spill, } T \in [0.01, 30.295]\,.$$

The simulator output is functional—the location × time concentration profile over $(s,t) \in (0,3) \times (0,60)$. For a spill made under conditions (M,D,L,T), let $y((s,t) \mid M,D,L,T)$ denote the concentration of the pollutant at (s,t). For example, Fig. 1.14 plots the concentration over $(s,t) \in (0,2.5) \times (0,60)$ when $(M,D,L,T) = (10, 0.07, 1.505, 30.1525)$. This figure shows that the first spill has little impact on concentrations for (downstream) locations s, $1.0 \le s \le 2.5$, and longer times t, $20 \le t \le 30.15$; the concentration is essentially back to pre-spill levels at these points. The impact of the spill at $(L,T) = (1.505, 30.1525)$ is more dramatic; the chemical diffuses to both sides of the spill location, increasing the concentration of the chemical and causing a noticeable increase in the concentration for times t, $30.15 \le t \le 60$. ♦

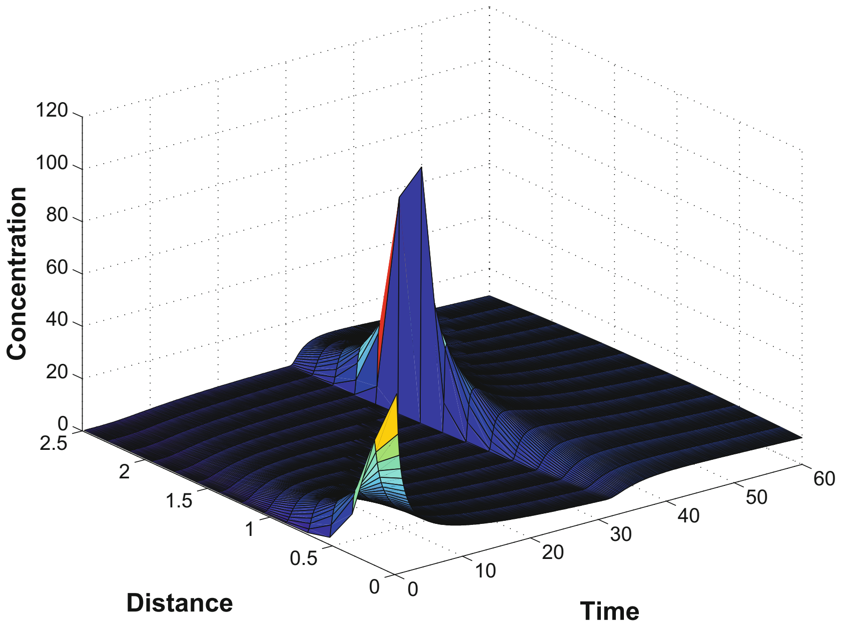


Fig. 1.14 Location, time, and concentration of a chemical in a channel having diffusion rate of 0.07 at locations s, $0 < s < 2.5$ and times t, $0 < t < 60$, resulting from a pair of spills each of mass 10 at $(s,t) = (0,0)$ and $(s,t) = (1.505, 30.1525)$

This subsection concludes with short sketches of several computer simulator models that involve *larger numbers of input variables* than the models described earlier in this section. These examples will also serve to broaden the reader's appreciation of the breath of scientific and engineering applications of simulator models.

Booker et al. (1997) describe an effort to design a helicopter blade with "optimal" geometry. While the main purpose of their report was the development of an optimization algorithm used to minimize a function of the computer model outputs, their application is of interest because the geometric specification of the rotor contained 31 input variables. The objective function was a measure of the rotor vibration that combined the forces and moments on the rotor. Each run of the computer simulator required very little time (10–15 min). However, the computer code provided a limited accuracy solution of the mathematical equations that describe the forces and moments on the rotor blade, than would a more detailed finite element code.

The rotor blade design application raises the broad question of how to combine runs of a fast-simulator code with those of slower but gold standard code for the same output. This issue will be addressed in Sect. 2.4. Chapter 8 will describe calibration methodology for combining information from multiple sources.

Lempert et al. (2002) describe a computer code used in public policy decision making. The objective of their research was to contrast the effects of several national policies for curbing the effect of greenhouse gases based on an "integrative" model of the future that links the world economy to the population and to the state of the environment. Lempert et al. (2002) utilized the so-called *Wonderland model* which quantifies the state of the future over a window of (typically) 100 years, using several output measures, one of which is a "human development index." The model is integrative in that, for example, the pollution at a given time point depends on the user-specified innovation rate for the pollution abatement, the current population, the output per capita, environmental taxes, and other factors. The human development index is a weighted average of a discounted annual improvement of four quantities including, for example, the (net) output per capita. In all, the simulator based on the Wonderland model has roughly 30 input variables; the inputs specify different public policies, initial world conditions, and evolution rates.

An additional source of motivating examples is Berk et al. (2002) who report on a workshop that discussed computer models in four diverse areas: transportation flows, wildfire evolution, the spatiotemporal evolution of storms, and the spread of infectious diseases.

1.3 Some Common Types of Computer Experiments

The previous section suggests the breadth of applications that employ computer simulator experiments. This section will describe three types of common simulator settings. To set notation, let $\boldsymbol{x}$ denote the simulator input, $\mathcal{X}$ the input domain, and $y(\boldsymbol{x})$ a *real-valued* simulator output. The first setting, called a *homogeneous-input* simulator, is one in which all inputs are of the *same type*, usually either control, environmental, *or* model variables. The second setting will describe some special cases of simulators having *mixed-input* $\boldsymbol{x}$. The third subsection describes several settings where simulators have multiple outputs.

1.3.1 Homogeneous-Input Simulators

First, suppose that $\boldsymbol{x}$ consists exclusively of control variables, i.e., $\boldsymbol{x} = \boldsymbol{x}_c$. In this case one important objective is to predict $y(\boldsymbol{x})$ "well" for all $\boldsymbol{x}$ in some subdomain of $\mathcal{X}$ so that alternative values of the control variables can be compared. One intuitive basis for judging a proposed predictor $\widehat{y}(\boldsymbol{x})$ is its *integrated squared error*

$$\int_{\mathcal{X}} \left[\widehat{y}(\boldsymbol{x}) - y(\boldsymbol{x})\right]^2 w(\boldsymbol{x})\, d\boldsymbol{x}, \tag{1.3.1}$$

where $w(\boldsymbol{x})$ is a nonnegative weight function that quantifies the importance of each $\boldsymbol{x} \in \mathcal{X}$. For example, $w(\boldsymbol{x}) = 1$ weights all parts of $\mathcal{X}$ equally. If $I\{\boldsymbol{x} \in \mathcal{A}\}$, $\mathcal{A} \subset \mathcal{X}$, is the function which equals 1 for $\boldsymbol{x} \in \mathcal{A}$ and is 0 otherwise, then $w(\boldsymbol{x}) = I\{\boldsymbol{x} \in \mathcal{A}\}$ ignores the complement of $\mathcal{A}$ and weights all points in $\mathcal{A}$ equally.

Unfortunately, (1.3.1) cannot be calculated because $y(\boldsymbol{x})$ is unknown. Later chapters will replace $\left[\widehat{y}(\boldsymbol{x}) - y(\boldsymbol{x})\right]^2$ by a posterior mean squared value computed under a certain prior model for $y(\boldsymbol{x})$ and obtain an approximation of (1.3.1) that can be computed.

The problem of predicting $y(\boldsymbol{x})$ well over a specified subregion of $\mathcal{X}$ can be thought of as a global objective. In contrast, most local goals focus on finding "interesting" parts of the input domain $\mathcal{X}$ where the importance of $\boldsymbol{x}$ depends on $y(\boldsymbol{x})$. An example of such a goal is to identify (any) $\boldsymbol{x}$, where $y(\boldsymbol{x})$ equals some target value. More formally suppose

$$\mathcal{L}(t_0) = \{\boldsymbol{x} \in \mathcal{X} \mid y(\boldsymbol{x}) = t_0\}$$

denotes the *level set* of all input values where $y(\boldsymbol{x})$ attains a given target value t_0. The objective stated above is that of finding any $\boldsymbol{x} \in \mathcal{L}(t_0)$. Another example of a local objective is to find extreme values of $y(\boldsymbol{x})$. The problem of finding any $\boldsymbol{x}$ in the set

$$\mathcal{M} = \left\{\boldsymbol{x} \in \mathcal{X} \;\middle|\; y(\boldsymbol{x}) \geq y(\boldsymbol{x}^\star) \text{ for all } \boldsymbol{x}^\star \in \mathcal{X}\right\}$$

is that of determining *any* input that attains the global maximum of $y(\boldsymbol{x})$. There is a large literature discussing solutions to problems of finding global optima of black box simulators (see Dixon and Szego (1978); Bernardo et al. (1992); Mockus et al. (1997); Mockus (1998); Jones et al. (1998); Trosset and Padula (2000); Regis and Shoemaker (2005); Vazquez and Bect (2010); Picheny et al. (2013); Gramacy et al. (2016) and the references therein).

As for control-only input simulators, there is a similarly large literature discussing problems when $\boldsymbol{x}$ depends only on *environmental variables*. Perhaps the simplest application is when the inputs have a *known* distribution representing input uncertainty, and the goal is to determine the probability distribution of the simulator output which quantifies the uncertainty in the response. Formally, let the upper case notation $\boldsymbol{x} = \boldsymbol{X}_e$ denote the distribution of the inputs when the objective is to find the distribution of $y(\boldsymbol{X}_e)$. This problem is sometimes called *uncertainty analysis* or *uncertainty quantification* (UQ) (Crick et al. (1988), Dandekar and Kirkendall (1993), Helton (1993), O'Hagan and Haylock (1997), and O'Hagan et al. (1999) provide examples of UQ). In the spirit of this formalization, McKay et al. (1979) introduced the class of Latin hypercube designs for choosing the training sites $\boldsymbol{X}_e$ at which to evaluate $y(\boldsymbol{x}_e)$ when the objective is to predict the *mean* of $y(\boldsymbol{X}_e)$, denoted by $E[y(\boldsymbol{X}_e)]$. The theoretical study of Latin hypercube designs has established a host of asymptotic and empirical properties of $E[y(\boldsymbol{X}_e)]$ estimators (Stein (1987); Owen (1992a, 1994); Loh (1996); Pebesma and Heuvelink (1999)) and enhancements of such designs (Handcock (1991); Tang (1993, 1994); Ye (1998); Ye et al. (2000); Butler (2001); Qian et al. (2006); Qian and Wu (2008); Qian (2009, 2012)).

A third case of homogeneous inputs is when $\boldsymbol{x}$ depends only on *model* variables, i.e., $\boldsymbol{x} = \boldsymbol{x}_m$. Occurring less frequently than the first two homogeneous input cases, simulators used in climate modeling simulators have this setup where the mathematical model involves possibly many *unknown* parameters. If in addition, there are data from the physical system modeled by the simulator, then calibrating the simulator is an important problem.

1.3.2 Mixed-Input Simulators

This subsection focusses on what is arguably the most interesting of the mixed-input cases. This is the situation when $\boldsymbol{x}$ consists of control and environmental inputs. The section assumes that $\boldsymbol{x} = (\boldsymbol{x}_c, \boldsymbol{X}_e)$ where, as in Sect. 1.3.1, $\boldsymbol{X}_e$ has a known distribution.

For each $\boldsymbol{x}_c$, $y(\boldsymbol{x}_c, \boldsymbol{X}_e)$ is a random variable with a distribution that is induced by the distribution of $\boldsymbol{X}_e$. This is a familiar regression-type setting although the model is more general. A simpler problem than determining the distribution of $y(\boldsymbol{x}_c, \boldsymbol{X}_e)$ is to merely estimate the mean of $y(\boldsymbol{x}_c, \boldsymbol{X}_e)$, which will be denoted by

$$\mu(\boldsymbol{x}_c) = E\left[y(\boldsymbol{x}_c, \boldsymbol{X}_e)\right]$$

where the expectation is with respect to the $\boldsymbol{X}_e$ distribution.

In some applications users would be interested in choosing the control inputs to, say, minimize $\mu(\boldsymbol{x}_c)$. To describe this and related goals in a formal fashion, denote the upper α quantile of the distribution of $y(\boldsymbol{x}_c, \boldsymbol{X}_e)$ by $\xi^\alpha = \xi^\alpha(\boldsymbol{x}_c)$ where

$$P\left\{y(\boldsymbol{x}_c, \boldsymbol{X}_e) \geq \xi^\alpha\right\} = \alpha$$

(assuming for simplicity that there is a unique such upper α quantile). For example, $\xi^{.5}(\boldsymbol{x}_c)$ denotes the *median* of the distribution of $y(\boldsymbol{x}_c, \boldsymbol{X}_e)$.

Consider analogs for $\mu(\boldsymbol{x}_c)$ of the three goals stated above for $y(\boldsymbol{x}_c)$. The analog of predicting $y(\boldsymbol{x}_c)$ well over the $\boldsymbol{x}_c$ domain is to predict $\mu(\boldsymbol{x}_c)$ well over the $\boldsymbol{x}_c$ domain in the sense of minimizing

$$\int \left[\mu(\boldsymbol{x}_c) - \widehat{\mu}(\boldsymbol{x}_c)\right]^2 \, w(\boldsymbol{x}_c)\, d\boldsymbol{x}_c \tag{1.3.2}$$

where $\widehat{\mu}(\boldsymbol{x}_c)$ denotes a generic predictor of $\mu(\boldsymbol{x}_c)$. To solve this problem, one must not only choose a particular method of constructing $\widehat{\mu}(\boldsymbol{x}_c)$ but also a set of "training sites" $(\boldsymbol{x}_c, \boldsymbol{x}_e)$ to estimate the specific $\widehat{\mu}(\boldsymbol{x}_c)$. As in the case of (1.3.1), the criterion (1.3.2) cannot be computed, but a Bayesian analog that has a computable mean will be introduced in Chap. 6. The parallel of the problem of finding a control variable to maximize $y(\boldsymbol{x}_c)$ is that of determining an $\boldsymbol{x}_c$ to maximize $\mu(\boldsymbol{x}_c)$, i.e., finding an $\boldsymbol{x}_c^M$ that satisfies

$$\mu(\boldsymbol{x}_c^M) \in \underset{\boldsymbol{x}_c}{\operatorname{argmax}}\ \mu(\boldsymbol{x}_c).$$

Similarly, the analog of the problem of finding $\boldsymbol{x}_c$ to attain a fixed target value of $y(\boldsymbol{x}_c)$ is straightforward to formulate for $\mu(\boldsymbol{x}_c)$. Of course, if the distribution of $y(\boldsymbol{x}_c, \boldsymbol{X}_e)$ is skewed, the objectives described below might better be stated in terms of $\xi^{.5}(\boldsymbol{x}_c)$.

Other problems are more relevant when the distribution of $\boldsymbol{X}_e$ is not known precisely. To illustrate, suppose that $\boldsymbol{x}_c^M$ maximizes $E_{G^N}[y(\boldsymbol{x}_c, \boldsymbol{X}_e)]$ for a given *nominal* $\boldsymbol{X}_e$ distribution, G^N. Suppose, instead, that $G \neq G^N$ *is the true* $\boldsymbol{X}_e$ *distribution*. If

$$E_G\left[y(\boldsymbol{x}_c^M, \boldsymbol{X}_e)\right] \ll \max_{\boldsymbol{x}_c} E_G\left[y(\boldsymbol{x}_c, \boldsymbol{X}_e)\right],$$

then $\boldsymbol{x}_c^M$ is (substantially) inferior to any $\boldsymbol{x}_c^\star$ that produces a larger mean value when $\boldsymbol{X}_e$ has distribution G, i.e., $E_G[y(\boldsymbol{x}_c^M, \boldsymbol{X}_e)] < E_G[y(\boldsymbol{x}_c^\star, \boldsymbol{X}_e)]$. From this perspective, a control variable $\boldsymbol{x}_c$ can be thought of as being "robust" against misspecification of the $\boldsymbol{X}_e$ distribution if $\boldsymbol{x}_c$ comes close to maximizing the mean over the nominal $\boldsymbol{X}_e$ distribution, and $\boldsymbol{x}_c$ is never far from achieving the maximum $E_G[y(\boldsymbol{x}_c^\star, \boldsymbol{X}_e)]$ for a specified set of alternative $\boldsymbol{X}_e$ distributions, G. There are several heuristic methods of defining a robust $\boldsymbol{x}_c$ that embody this idea.

One of the earliest methods of defining a robust $\boldsymbol{x}_c$ uses minimaxity (Huber (1981)). Given a set $\mathcal{G}$ of possible $\boldsymbol{X}_e$ distributions that includes the "central" nominal distribution G^N, let

$$\mu(\boldsymbol{x}_c, G) = E_G\left[y(\boldsymbol{x}_c, \boldsymbol{X}_e)\right]$$

denote the mean of $y(\boldsymbol{x}_c, \boldsymbol{X}_e)$ when $\boldsymbol{X}_e$ has distribution $G \in \mathcal{G}$. Then

$$\min_{G \in \mathcal{G}} \mu(\boldsymbol{x}_c, G)$$

is the smallest mean value for $y(\boldsymbol{x}_c, \boldsymbol{X}_e)$ that is possible when $\boldsymbol{X}_e$ distributions come from $\mathcal{G}$. We say $\boldsymbol{x}_c^{\mathcal{G}}$ is $\mathcal{G}$*-robust* provided

$$\min_{G \in \mathcal{G}} \mu(\boldsymbol{x}_c^{\mathcal{G}}, G) = \max_{\boldsymbol{x}_c} \min_{G \in \mathcal{G}} \mu(\boldsymbol{x}_c, G).$$

However mathematically satisfying their definition, $\mathcal{G}$-robust choices of $\boldsymbol{x}_c$ can be criticized on at least three grounds. They are *pessimistic*; $\boldsymbol{x}_c^{\mathcal{G}}$ maximizes a worst-case scenario for the mean of $y(\boldsymbol{x}_c, \boldsymbol{X}_e)$. Specifying a meaningful $\mathcal{G}$ for a given application can be *arbitrary*. Finally, there can be substantial *computational* difficulties determining $\mathcal{G}$-robust $\boldsymbol{x}_c$.

Lehman et al. (2004) give alternative definitions of robustness and provide sequential designs for identifying robust $\boldsymbol{x}_c$.

1.3.3 Multiple Outputs

To fix ideas, suppose that $y_1(\cdot), \ldots, y_m(\cdot)$ are the simulator outputs. This subsection describes four of the many settings that lead to such data and formulates several common problems that occur in multiple output settings.

In some cases, the outputs can be multiple simulators of the *same* physical quantity; for example, Kennedy and O'Hagan (2000) study simulators based on coarser and finer FE grids for the same response. In other cases the simulators could implement more or less detailed mathematical models describing the relationship between the input/output variables.

A second situation that leads to multiple outputs is when $y_1(\cdot), \ldots, y_m(\cdot)$ are *competing* responses. For example, the design of an airplane wing should both maximize lift and minimize drag. A car is designed to minimize body weight (the single most important factor in determining gas mileage) and maximize body strength (to increase safety in a crash).

A third setting that leads to multiple outputs is when $y_1(\cdot), \ldots, y_m(\cdot)$ each contain partial information about a common output. Arguably the most important example of such a situation is described in Morris et al. (1993) and Mitchell et al. (1994) who consider the prediction of a given $y(\boldsymbol{x})$ for cases where the simulator provides $y(\boldsymbol{x})$ *and* all first partial derivatives of $y(\boldsymbol{x})$. Regarding $y_1(\boldsymbol{x}) = y(\boldsymbol{x})$ and $y_2(\boldsymbol{x}), \ldots, y_m(\boldsymbol{x})$ as the values of the partial derivatives of $y(\boldsymbol{x})$ with respect to each component of $\boldsymbol{x}$ produces the multiple outputs for each $\boldsymbol{x}$. Intuitively, the partial derivatives should provide auxiliary information that permits more precise prediction of future $y(\boldsymbol{x})$ values than predictions based on $y(\boldsymbol{x})$ training data alone.

A final setting that produces multiple outputs is that of discretized functional output. To fix ideas consider the simple case when the function output depends on time, and for each input $\boldsymbol{y}(\boldsymbol{x}) \equiv (y(\boldsymbol{x}; t_1), \ldots, y(\boldsymbol{x}; t_m))$ is a vector of m values at the time points $t_1, \ldots, t_m$. More general spatial or space $\times$ time outputs follow a similar pattern. While smooth functional data can appear to be high dimensional when m is large, the $m \times n$ data can often be reduced to a lower-dimensional representation as will be seen in Chap. 8 during the discussion of model parameter calibration.

Depending on the nature of the simulator output, many scientific objectives can be of interest in multivariate simulator data settings. Consider the scenario where $\boldsymbol{x} = \boldsymbol{x}_c$, $y_1(\cdot)$ is the response of primary interest which is to be *minimized*, and $y_2(\cdot), \ldots, y_m(\cdot)$ are competing objectives which are to be *maximized*. An approximate solution can be computed by solving the following constrained optimization problem. First construct a feasible region of $\boldsymbol{x}_c$ values by setting minimal (user-specified) performance standards for $y_2(\boldsymbol{x}_c), \ldots, y_m(\boldsymbol{x}_c)$. Then solve

$$\text{minimize } y_1(\boldsymbol{x}_c)$$

$$\begin{aligned} &\text{subject to} \\ &y_2(\boldsymbol{x}_c) \geq M_2 \\ &\quad\vdots \\ &y_m(\boldsymbol{x}_c) \geq M_m\,. \end{aligned} \tag{1.3.3}$$

Here M_i denotes the lowest acceptable bound on the performance of $y_i(\cdot)$, $i = 2, \ldots, m$.

If, in addition to control variables, $\boldsymbol{x}$ contains environmental variables $\boldsymbol{X}_e$, then each $y_i(\boldsymbol{x}_c)$ in (1.3.3) can be replaced by $\mu_i(\boldsymbol{x}_c) = E[y_i(\boldsymbol{x}_c, \boldsymbol{X}_e)]$. Lastly, if $y_1(\boldsymbol{x}), \ldots, y_m(\boldsymbol{x})$ are the outputs of *different* codes of varying accuracy for the same response, then a typical goal would be to combine information from the various outputs to better predict the true response or to predict the code that is regarded as the most detailed. Further discussion of this idea is postponed until modeling multiple response outputs is discussed in Sects. 2.5 and 3.5.

1.4 Organization of the Remainder of the Book

Before providing an overview of the book chapters, we describe the *Notes* sections that conclude each chapter. Notes sections give an overview of some of the developments that extend the basic ones explained in this book. They list additional references and give details for some of the topics that are sketched in the chapter. Finally Notes sections list, as of the publication of this volume and consistent with the authors' knowledge and experience, software that is available to carry out the analyses described in the chapter.

The remainder of the book is organized as follows. Chapter 2 describes the stochastic process interpolating models that form the basis for most of the design and analysis methodology that is presented in Chaps. 3–8. Both stationary models and nonstationary models are included. The chapter develops intuition by showing functions $y(\boldsymbol{x})$ drawn from these models. Finally, it introduces process models for multivariate output.

Using training data runs, Chap. 3 describes empirical best linear unbiased prediction (EBLUP) methodology based on the Gaussian process model for predicting simulator output at new inputs. It also presents methods for assessing uncertainty in the predictions. The chapter compares EBLUP methods via simulation and presents our recommended choice of predictor.

Chapter 4 parallels Chap. 3 in describing a fully Bayesian predictor of simulator output. Both conjugate and non-conjugate cases are presented where the latter must be implemented by appropriate sampling from the posterior.

Chapters 5 and 6 describe a wide variety of experimental designs for computer simulator experiments, i.e., input sites at which to run code. Chapter 5 considers *space-filling* designs, meaning designs that "spread" observations throughout the input region. Among the designs examined are those based on simple random sampling, Latin hypercube designs, orthogonal arrays, distance-based designs, uniform designs, and designs that combine multiple criteria. Grid and lattice designs are presented briefly at the end of Chap. 5. Chapter 6 considers designs based on *statistical* criteria; in most cases approximations of the criteria depend on the availability of a suitable interpolating process. The criteria include maximum entropy and mean squared error of prediction. Chapter 6 also considers sequential strategies for designing computer experiments to optimize simulator output and for other objectives.

Chapter 7 presents an introduction to tools for sensitivity analysis and the related topic of variable screening. Sensitivity analysis methodology seeks to determine for each input, whether that input is "active" or not. Both graphical and numerical methods are presented. Variable screening is a decision procedure for determining the subset of active inputs.

Finally, Chap. 8 discusses problems that occur when both simulator and physical system data are available. These problems include inferring the calibration parameters of the simulator code based on the combined data and quantifying the uncertainty in their values. Another important objective that is described in this chapter is to combine simulator code and physical system data to predict the *mean of the physical system* output and to quantify the uncertainty of the prediction. Chapter 8 also extends the Bayesian prediction of Chap. 4 to cases of multivariate and functional simulator output.

Chapter 2
Stochastic Process Models for Describing Computer Simulator Output

2.1 Introduction

Recall from Chap. 1 that $\boldsymbol{x}$ denotes a generic input to our computer simulator and $y(\boldsymbol{x})$ denotes the associated output. This chapter will introduce several classes of random function models for $y(\boldsymbol{x})$ that will serve as the core building blocks for the interpolators, experimental designs, calibration, and tuning methodologies that will be introduced in later chapters. The reason that the random function approach is so useful is that accurate prediction based on black box computer simulator output requires a rich class of $y(\boldsymbol{x})$ options when only a minimal amount might be known about the output function. Indeed, regression mean modeling of simulator output is usually based on a rather arbitrarily selected parametric form.

While some readers will regard the process model as an extension of the familiar regression model, others will see it as a Bayesian (prior) model for $y(\boldsymbol{x})$. The process model can be constructed to ensure smoothness and monotonicity features of $y(\boldsymbol{x})$ that are known before data are collected, and hence the process is a prior for $y(\boldsymbol{x})$ (see Oakley (2002) and Reese et al. (2004) for advice about eliciting prior information and case studies about the formation of prior distributions for simulator models and more generally Berger (1985) and O'Hagan (1994)).

The viewpoint taken in this book is Bayesian because the authors find this to be the most philosophically satisfying approach, for example, the Bayesian interpretation of an interval estimate of a predicted value describes the uncertainty in $y(\boldsymbol{x})$ given the training data. Unfortunately, it is not always possible to elicit informative prior information. Thus our approach is not dogmatic; while the process model controls the characteristics of the functions produced by our priors, our attitude is to not *rigidly* believe them. Our approach is to choose flexible priors that are capable of producing many shapes for $y(\cdot)$ and then let the Bayesian machinery direct the details of the prediction and other inference processes.

The design and analysis of computer experiments are not the only statistical disciplines to use Bayesian methodology to analyze highly correlated $y(\boldsymbol{x})$ data, often measured with error. Examples of scientific disciplines that produce such data

T. J. Santner et al., *The Design and Analysis of Computer Experiments*,
Springer Series in Statistics, https://doi.org/10.1007/978-1-4939-8847-1_2

are geostatistics (Matheron (1963), Journel and Huijbregts (1978)), environmental statistics and disease mapping (Ripley (1981), Cressie (1993)), global optimization (Mockus et al. (1997)), and statistical learning (Hastie et al. (2001)). Hence many of the methodologies discussed in the literatures of these disciplines are relevant for computer experiments.

In the following let $\mathcal{X}$ denote the input space for the unknown output $y(\boldsymbol{x})$. The function $y(\cdot)$ is regarded as a draw from a random function ("stochastic process" or simply "process") which is denoted by $Y(\cdot)$. This book will adopt a pragmatic viewpoint in discussing stochastic process models rather than a measure-theoretic one. However, it will enhance our understanding of the subject to know that a random function is best thought of as a mapping from a sample space of elementary outcomes, say Ω, to a given set of functions just as a random variable is a mapping from a set of outcomes Ω to the real numbers. It will occasionally add clarity to a discussion to write $y(\boldsymbol{x}) = Y(\boldsymbol{x}, \omega)$, $\omega \in \Omega$, recognizing that $Y(\cdot, \omega)$ refers to a *particular* function from $\mathcal{X}$ to $\mathbb{R}^1$. Thus $y(\boldsymbol{x}) = Y(\boldsymbol{x}, \omega)$ is referred to as a *draw* or *realization* of the random function $Y(\omega)$. One place where the introduction of the underlying sample space Ω helps clarify ideas is when discussing the smoothness properties of functions $y(\boldsymbol{x}) = Y(\boldsymbol{x}, \omega)$ drawn from a process.

To introduce the idea of a random function to readers who are not familiar with this notion, our introduction is concluded with an example that illustrates a simple method for generating a random quadratic function.

Example 2.1. Suppose $y(x)$ on $[-1, +1]$ is drawn from the mechanism

$$Y(x) = b_0 + b_1 x + b_2 x^2 \,, \tag{2.1.1}$$

where b_0, b_1, and b_2 are mutually independent with $b_i \sim N(0, \sigma_i^2)$ for $i = 0, 1, 2$. Functions drawn from $Y(x)$ are simple to visualize. Every realization $y(\cdot)$ is a quadratic equation ($P[b_2 = 0] = 0$) that is symmetric about an axis other than the y-axis (symmetry about the y-axis occurs if and only if $b_1 = 0$ but $P[b_1 = 0] = 0$). The quadratic is convex with probability 1/2 and it is concave with probability 1/2 (because $P[b_2 > 0] = 1/2 = P[b_2 < 0]$). Figure 2.1 illustrates ten outcomes from $Y(x)$ when $\sigma_0^2 = \sigma_1^2 = \sigma_2^2 = 1.0$.

For any $x \in [-1, +1]$ the draws from (2.1.1) have mean zero, i.e.,

$$\begin{aligned} E\left[Y(x)\right] &= E\left[b_0 + b_1 x + b_2 x^2\right] \\ &= E\left[b_0\right] + E\left[b_1\right] \times x + E\left[b_2\right] \times x^2 \\ &= 0 + 0 \times x + 0 \times x^2 = 0\,. \end{aligned} \tag{2.1.2}$$

Equation (2.1.2) says that for any x, the mean of $Y(x)$ is *zero* over many drawings of the coefficients (b_0, b_1, b_2). Likewise for any $x \in [-1, +1]$ the variance of $Y(x)$ is

$$\begin{aligned} Var\left[Y(x)\right] &= E\left[\left(b_0 + b_1 x + b_2 x^2\right)\left(b_0 + b_1 x + b_2 x^2\right)\right] \\ &= \sigma_0^2 + \sigma_1^2 x^2 + \sigma_2^2 x^4 \geq 0\,. \end{aligned} \tag{2.1.3}$$

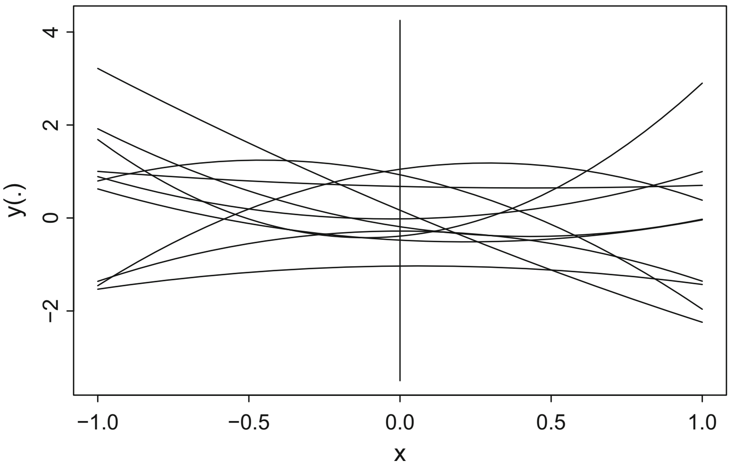


Fig. 2.1 Ten draws from the random function $Y(x) = b_0 + b_1 x + b_2 x^2$ on $[-1, +1]$, where b_0, b_1, and b_2 are independent and identically $N(0, 1.0)$ distributed

Both features (2.1.2) and (2.1.3) can be seen in Fig. 2.1. For example, the effect of (2.1.3) is seen in the greater spread in the $y(-1)$ and $y(+1)$ values compared with the spread in the $y(0)$ values after only ten function draws.

Additionally, the values of $Y(x_1)$ and $Y(x_2)$ at $x_1, x_2 \in [-1, +1]$ are related, as can be seen from

$$\begin{aligned} Cov\,[Y(x_1), Y(x_2)] &= E\left[\left(b_0 + b_1 x_1 + b_2 x_1^2\right)\left(b_0 + b_1 x_2 + b_2 x_2^2\right)\right] \\ &= \sigma_0^2 + \sigma_1^2 x_1 x_2 + \sigma_2^2 x_1^2 x_2^2 \,. \end{aligned} \tag{2.1.4}$$

The covariance (2.1.4) can be positive or negative. The sign of the covariance of $Y(x_1)$ and $Y(x_2)$ can intuitively be explained as follows. The covariance is positive for any x_1 and x_2 that are *both positive* or *both negative*, i.e., on the same side of the y axis, the vertical line that passes through $x = 0$. Intuitively this is true for any such x_1, x_2 because over many drawings of (b_0, b_1, b_2), x_1 and x_2 both tend to be on the same side of the axis of symmetry of $Y(x)$ and thus $Y(x_1)$ and $Y(x_2)$ increase or decrease together (see Fig. 2.1). The covariance formula can be negative if x_1 and x_2 are on the *opposite* sides of the origin so that the middle term in (2.1.4) is negative *and* its value dominates the sum of the positive terms σ_0^2 and $\sigma_2^2 x_1^2 x_2^2$. Intuitively, one circumstance where this occurs is if σ_0^2 is small (meaning the curves tend to fall "near" the point $(0, 0)$), *and* σ_2^2 is small (the curves are near linear for x close to 0), *and* σ_1^2 is large. In this case, the draws fluctuate between those with large positive slopes and those with large negative slopes, implying that $Y(x_1)$ and $Y(x_2)$ tend to have the opposite sign over many draws.

Because linear combinations of a fixed set of independent normal random variables have the multivariate normal distribution, the $Y(x)$ model (2.1.1) satisfies the following property. For each $L > 1$ and any choice of $x_1, \ldots, x_L \in \mathcal{X}$, the vector $(Y(x_1), \ldots, Y(x_L))$ is multivariate normally distributed. (See Appendix B for a review of the multivariate normal distribution.)

The $y(\cdot)$ realizations have several critical limitations from the viewpoint of describing computer simulator output. First, $Y(x)$ can *only* produce quadratic draws. Second, the multivariate normal distribution of $(Y(x_1), \ldots, Y(x_L))$ is *degenerate* when $L \geq 4$. The development in the remainder of the chapter provides more flexible random functions that retain the computational advantage that $(Y(x_1), \ldots, Y(x_L))$ has a nondegenerate multivariate normal distribution. ♦

The remainder of this chapter is organized as follows. Sections 2.2–2.4 discuss models for real-valued outputs. Section 2.2 reviews the frequently used class of Gaussian process (GP) models, Sect. 2.3 discusses some nonstationary extensions of the GP model, and Sect. 2.4 presents models for simulators having mixed quantitative and qualitative inputs. Section 2.5 describes models for simulators that produce multivariate or functional output.

2.2 Gaussian Process Models for Real-Valued Output

2.2.1 Introduction

Because of their analytical tractability, the most popular $Y(\boldsymbol{x})$ processes for generating function draws in the computer experiments literature are *Gaussian process* (GP) models, also called *Gaussian random function* models. In addition, mixtures of GPs are used in Bayesian modeling, as considered in Chap. 4. Hence this section emphasizes GP models although, as will be noted, some of the concepts that are introduced apply to more general random function models.

Definition. Suppose that $\mathcal{X}$ is a fixed subset of $\mathbb{R}^d$ having positive d-dimensional volume; $Y(\boldsymbol{x})$, $\boldsymbol{x} \in \mathcal{X}$, is a GP provided that for any $L \geq 1$ and any choice of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L$ in $\mathcal{X}$, the vector $(Y(\boldsymbol{x}_1), \ldots, Y(\boldsymbol{x}_L))$ has a multivariate normal distribution.

Any GP is determined by its *mean* function, $\mu(\boldsymbol{x}) \equiv E[Y(\boldsymbol{x})]$, $\boldsymbol{x} \in \mathcal{X}$, and by its *covariance* function

$$C(\boldsymbol{x}_1, \boldsymbol{x}_2) \equiv Cov\,[Y(\boldsymbol{x}_1), Y(\boldsymbol{x}_2)]\,, \tag{2.2.1}$$

for $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$. To be consistent with the language used in time series analysis, some authors call $C(\cdot, \cdot)$ the "autocovariance" function.

The GPs that are used in practice are *nonsingular*, which means that for any choice of inputs, the covariance matrix of the associated multivariate normal distribution is nonsingular. Nonsingular multivariate normal distributions have the advantage that it is easy to compute the conditional distribution of one (or several) of the $Y(\boldsymbol{x}_i)$ variables given the remaining $Y(\boldsymbol{x}_j)$. The empirical best linear unbiased prediction methodology used in Chap. 3 requires that these conditional means and conditional variances be known, and the fully Bayesian predictive distributions of Chap. 4 require that the entire $Y(\boldsymbol{x})$ conditional distribution be known. In addition, draws from the most widely used GPs allow a greater spectrum of shapes than the quadratic equations generated in Example 2.1. They also permit the modeler to control the smoothness properties of the $y(\boldsymbol{x})$ draws; in most of the scientific applications mentioned above, there is *some* information about the smoothness of $y(\cdot)$, although perhaps only that it is a continuous function of the inputs.

There are two technical issues that will be addressed before introducing specific GP models. The first issue has to do with the fact that GP models are defined by their *finite-dimensional* distributions. In contrast, smoothness properties such as continuity or differentiability depend on limiting operations and hence on knowledge that the draw behaves in a desired fashion over an *interval* of values. In other words, continuity and differentiability of $y(\boldsymbol{x})$ as a function of $\boldsymbol{x}$ are *sample path properties*, i.e., they regard $y(\boldsymbol{x}) = Y(\boldsymbol{x}, \omega)$ as a function of $\boldsymbol{x}$ for fixed ω. Doob (1953) introduced a property of stochastic processes called *separability*, which ensures that that sample path properties of function draws are determined by finite-dimensional distributions of the process. While the details of the exact meaning of separability are outside the scope of this book, interested readers can consult Adler (1981) who gives a mathematical definition of this property and additional discussion of its intuition (page 15). For our purposes it suffices to know that given any random function $Y(\cdot)$ on $\mathcal{X}$, there is an equivalent separable random function $Y^s(\cdot)$ on $\mathcal{X}$ such that

$$P\left[Y(\boldsymbol{x}) = Y^s(\boldsymbol{x})\right] = 1 \;\text{ for all }\; \boldsymbol{x} \in \mathcal{X}.$$

Throughout this book, it is assumed that GP models have been chosen to be separable.

The second technical issue is statistical. Classical frequentist statistical methods make inferences about a population based on sample draws from that population, where it is to be emphasized that each draw corresponds to a different ω or member of the population. Then statistical procedures are devised to have specified properties in hypothetical, repeated applications to the same population, e.g., the coverage probability of a prediction interval refers to the performance of the statistical procedure in repeated applications to a target population. In contrast, the "training data" $y(\boldsymbol{x}_1), \ldots, y(\boldsymbol{x}_n)$ from n runs of a computer simulator are properly viewed as the values of $y(\boldsymbol{x}) = Y(\boldsymbol{x}, \omega)$ corresponding to a *single* ω. Thus training data gives *partial* information about a single function $Y(\boldsymbol{x}, \omega)$, its value at $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, rather than the values of n random variables corresponding to n different ω. In general, it need not be the case that one can make inference about ω-defined population quantities from data that come from a single ω. *Ergodicity* of a process is a property that permits valid frequentist statistical inference of ω-defined quantities based on a single ω draw (for a discussion of ergodicity from a statistical viewpoint, see Cressie (1993), pages 52–58, and the additional references listed there). The technical details of this concept are beyond the scope of this book. However, GPs that are "(strongly) *stationary*," a stochastic repeatability which is defined next, are ergodic under a mild condition which is stated below; attention will be restricted to such GPs throughout the remainder of the text.

Definition. The process $Y(\cdot)$ is (*strongly*) *stationary* provided that for any $\boldsymbol{h} \in \mathbb{R}^d$, any $L \geq 1$, and any $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L$ in $\mathcal{X}$ having $\boldsymbol{x}_1 + \boldsymbol{h}, \ldots, \boldsymbol{x}_L + \boldsymbol{h} \in \mathcal{X}$, $(Y(\boldsymbol{x}_1), \ldots, Y(\boldsymbol{x}_L))$ and $(Y(\boldsymbol{x}_1 + \boldsymbol{h}), \ldots, Y(\boldsymbol{x}_L + \boldsymbol{h}))$ have the *same* distribution.

When applied to GPs, the stationarity of $Y(\cdot)$ is equivalent to requiring that $(Y(\boldsymbol{x}_1), \ldots, Y(\boldsymbol{x}_L))$ and $(Y(\boldsymbol{x}_1 + \boldsymbol{h}), \ldots, Y(\boldsymbol{x}_L + \boldsymbol{h}))$ have the same mean vector and same covariance matrix for any $L \geq 1$ and $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L$. In particular, GPs must have the *same* marginal distribution for all $\boldsymbol{x}$ (taking $L = 1$); their mean and their variance must both be constant. Also, it is not difficult to show that the covariance of a stationary GP must satisfy

$$Cov\left[Y(\boldsymbol{x}_1), Y(\boldsymbol{x}_2)\right] = C\left(\boldsymbol{x}_1 - \boldsymbol{x}_2\right) \tag{2.2.2}$$

for a function $C(\cdot)$, called the *covariance function* of the process. This is a slight abuse of the notation introduced in (2.2.1), because the function $C(\boldsymbol{x}_1, \boldsymbol{x}_2)$ defined in (2.2.1) for arbitrary GPs need not depend on the difference $\boldsymbol{x}_1 - \boldsymbol{x}_2$, whereas covariance of a stationary GP depends only on the difference between the inputs.

The (constant) variance of a stationary process $Y(\boldsymbol{x})$ can be expressed in terms of its covariance function as $Var[Y(\boldsymbol{x})] = Cov[Y(\boldsymbol{x}), Y(\boldsymbol{x})] = C(\boldsymbol{0})$. Putting these two facts together gives the following expression for the *correlation* of the stationary $Y(\boldsymbol{x})$,

$$Cor\left[Y(\boldsymbol{x}_1), Y(\boldsymbol{x}_2)\right] = C\left(\boldsymbol{x}_1 - \boldsymbol{x}_2\right) / C(\boldsymbol{0})\,.$$

Equation (2.2.2) means that all pairs $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ having common orientation *and* the same interpoint distance will have the same covariance. For example, the three head-and-tail pairs shown in the left-hand panel of Fig. 2.2 have the same covariance.

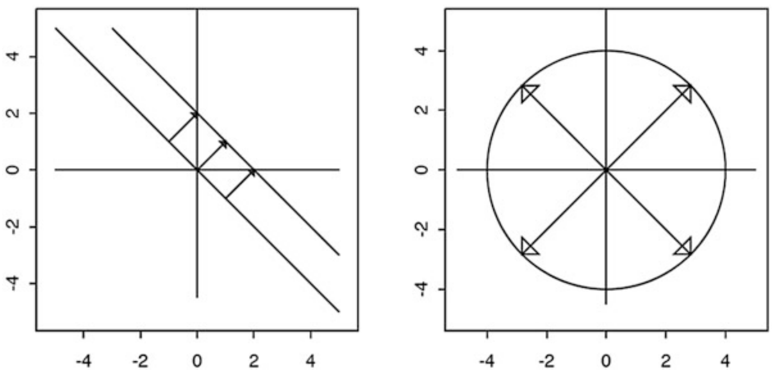


Fig. 2.2 Left panel: the tip and tail of each arrow have the same covariance for stationary processes. Right panel: the origin and every point on the circle have the same correlation for isotropic processes

Figure 2.3 illustrates the stochastic similarity of the five draws from two different stationary $Y(\boldsymbol{x})$; the behavior of $y(\boldsymbol{x})$ over any subinterval of a given length is the same as over any other disjoint subinterval of the same length. As a caution, the reader is warned that visually many $y(\boldsymbol{x})$ do not "appear" to be consistent with a stationary $Y(\boldsymbol{x})$ model. For example, the $y(x_1, x_2)$ in Fig. 2.4 behaves differently on the periphery of (x_1, x_2) space than in the center of the input space.

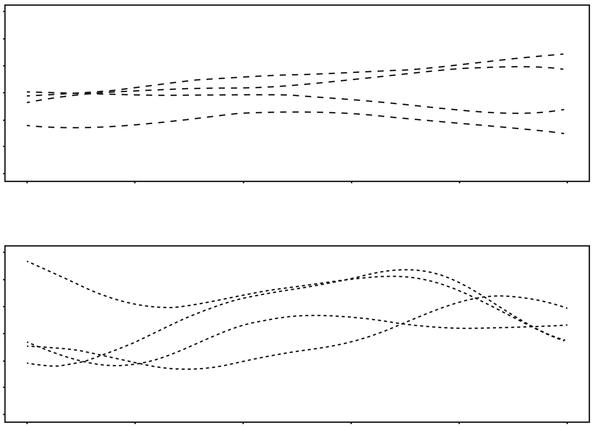


Fig. 2.3 Five sets of draws from each of the two stationary processes

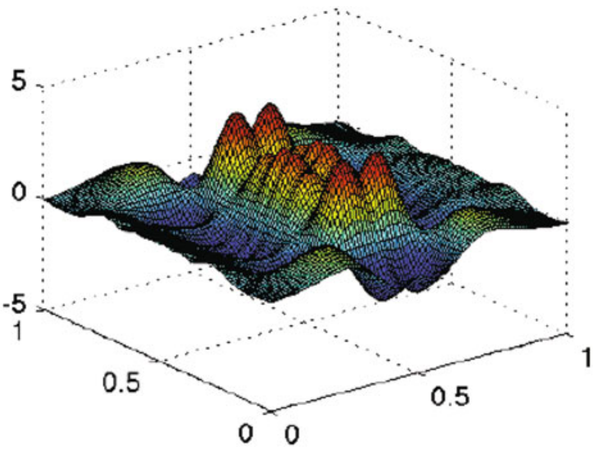


Fig. 2.4 A function $y(x_1, x_2)$ that visually appears to be inconsistent with a stationary model

A stationarity GP $Y(\boldsymbol{x})$ with covariance function $C(\boldsymbol{h})$ will be ergodic provided $C(\boldsymbol{h}) \to 0$ as $\boldsymbol{h} \to \infty$ (Adler (1981), page 145). The correlation function examples below satisfy this condition.

For completeness, our discussion of stochastic repeatability, i.e., stationarity, is concluded by describing a stronger form of stationarity that is assumed frequently in applications of spatial statistics but seldom in modeling computer simulator output.

Suppose, possibly after transforming the inputs $\boldsymbol{x}$, that $Y(\boldsymbol{x})$ has constant mean and covariance of the form

$$Cov\,[Y(\boldsymbol{x}_1), Y(\boldsymbol{x}_2)] = C\,(\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2)\,,$$

where $\|\boldsymbol{h}\|_2 = (\sum_i h_i^2)^{1/2}$ is Euclidean distance; such a process is said to be *isotropic*. For isotropic models, every pair of points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ having common interpoint distance must have the same covariance (and correlation) *regardless of their orientation*. For example, the origin and every point on the circle shown in the right-hand panel of Fig. 2.2 would all have the same correlation for an isotropic $Y(\boldsymbol{x})$. Because the inputs to simulator models are usually measured on different scales, isotropic models have been used infrequently to describe simulator output.

Later chapters will occasionally consider process models $Y(\cdot)$ that make only moment rather than distributional assumptions (and hence should be thought of as nonparametric). The most important such model for computer experiments is the *second-order stationary* model. A process $Y(\boldsymbol{x})$ having *constant mean* and *constant variance* is second-order stationary provided its covariance function satisfies (2.2.2); no assumption is made about the joint distribution of $Y(\boldsymbol{x})$ at any single or finite set of inputs;

Several approaches have been used in the literature to enhance random function modeling while retaining (some) of the theoretical simplifications that stationarity provides. The simplest of these techniques is to permit the mean of the stochastic process generating $y(\boldsymbol{x})$ to depend on $\boldsymbol{x}$ in the form of a regression equation while assuming the residual variation follows a stationary GP. The corresponding process has the form

$$Y(\boldsymbol{x}) = \sum_{j=1}^{p} f_j(\boldsymbol{x})\beta_j + Z(\boldsymbol{x}) = \boldsymbol{f}^\top(\boldsymbol{x})\boldsymbol{\beta} + Z(\boldsymbol{x}), \tag{2.2.3}$$

where $f_1(\cdot), \ldots, f_p(\cdot)$ are *known* regression functions, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^\top$ is a vector of *unknown* regression coefficients and $Z(\cdot)$ is a *zero mean* stationary GP over $\mathcal{X}$. Intuitively, the term $\boldsymbol{f}^\top(\boldsymbol{x})\boldsymbol{\beta}$ describes long-term trends in $\boldsymbol{x}$, while $Z(\boldsymbol{x})$ models local deviations from the long-term trend.

The $Y(\cdot)$ model (2.2.3) is, of course, nonstationary. There has been considerable development of alternative nonstationary models for $y(\boldsymbol{x})$ which cannot be described as a draw from Eq. (2.2.3). A brief review of some of this literature is provided in Sect. 2.3.

2.2.2 Some Correlation Functions for GP Models

This subsection focuses on the GP, $Z(\cdot)$, that is introduced in the nonstationary $Y(\boldsymbol{x})$ model in Eq. (2.2.3). Again note that $Z(\cdot)$ is stationary with *mean zero* because all nonzero mean terms are included in the regression function. Thus $Z(\cdot)$ is completely determined by its covariance function $C(\cdot)$ which is defined in (2.2.2).

In many applications, it is more convenient to separately describe the process variance and process *correlation function*. For a stationary GP $Z(\boldsymbol{x})$ having finite covariance function $C(\cdot)$, the process variance is $\sigma_Z^2 \equiv C(\mathbf{0})$ and the process *correlation function* is defined to be

$$R(\boldsymbol{h}) = C(\boldsymbol{h})/\sigma_Z^2 \;\text{ for }\; \boldsymbol{h} \in \mathbb{R}^d$$

assuming $Z(\boldsymbol{x})$ is nondegenerate, i.e., $C(\mathbf{0}) = \sigma_Z^2 > 0$. Again assuming that $\sigma_Z^2 > 0$, the process variance is sometimes more conveniently parameterized by the *process precision* which is defined by $\lambda_Z = 1/\sigma_Z^2$.

What properties must valid covariance and correlation functions of stationary GP processes possess, assuming $C(\mathbf{0}) > 0$? First, $R(\mathbf{0}) = 1$ follows by definition. Because $Cov[Y(\boldsymbol{x}+\boldsymbol{h}), Y(\boldsymbol{x})] = Cov[Y(\boldsymbol{x}), Y(\boldsymbol{x}+\boldsymbol{h})]$, both the covariance and correlation functions must be *symmetric about the origin*, i.e.,

$$C(\boldsymbol{h}) = C(-\boldsymbol{h}) \;\text{ and }\; R(\boldsymbol{h}) = R(-\boldsymbol{h})\,. \tag{2.2.4}$$

Both $C(\cdot)$ and $R(\cdot)$ must be *positive semidefinite* functions; stated in terms of $R(\cdot)$, this means that for any $L \geq 1$, and any real numbers $w_1, \ldots, w_L$, and any inputs $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L$ in $\mathcal{X}$,

$$\sum_{i=1}^{L}\sum_{j=1}^{L} w_i w_j\, R(\boldsymbol{x}_i - \boldsymbol{x}_j) \geq 0. \tag{2.2.5}$$

The sum (2.2.5) must be nonnegative for a valid $R(\cdot)$ ($C(\cdot)$) because the left-hand side is the (scaled) variance of $\sum_{i=1}^{L} w_i Y(\boldsymbol{x}_i)$. The correlation function $R(\cdot)$ is *positive definite* provided >0 holds in (2.2.5) for any $L \geq 1$, any $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L$ in $\mathcal{X}$, and any $(w_1, \ldots, w_L) \neq \mathbf{0}$.

How can one produce valid correlation (covariance) functions? Every function that satisfies $R(\mathbf{0}) = 1$ and the symmetry properties (2.2.4) and (2.2.5) is a valid correlation function. Unfortunately, merely requiring these properties does not offer a systematic method for constructing correlation functions. While a general study of how to determine the form of valid stationary correlation functions is beyond the scope of this book, one answer to this question is relatively simple to state. Bochner (1955) showed that

$$R(\boldsymbol{h}) = \int_{\mathbb{R}^d} \cos(\boldsymbol{h}^\top \boldsymbol{w}) f(\boldsymbol{w})\, d\boldsymbol{w} \tag{2.2.6}$$

is a valid correlation function provided that $f(\boldsymbol{w})$ is a symmetric density on $\mathbb{R}^d$, i.e., $f(\boldsymbol{w}) = f(-\boldsymbol{w})$ for all $\boldsymbol{w} \in \mathbb{R}^d$. The $f(\boldsymbol{w})$ in (2.2.6) is called the *spectral density* corresponding to $R(\boldsymbol{h})$.

It is straightforward to show that functions $R(\boldsymbol{h})$ constructed using (2.2.6) satisfy $R(\mathbf{0}) = 1$ and the properties (2.2.4) and (2.2.5). For example,

$$R(\mathbf{0}) = \int_{\mathbb{R}^d} \cos(0)\, f(\boldsymbol{w})\, d\boldsymbol{w} = 1,$$

because $f(\boldsymbol{w})$ is a density function. Symmetry (2.2.4) holds because $\cos(-x) = \cos(x)$ for all real x. Positive semidefiniteness (2.2.5) is true because for any $L \geq 1$, any real numbers $w_1, \ldots, w_L$, and any $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L$,

$$\begin{aligned}
&\sum_{i=1}^{L}\sum_{j=1}^{L} w_i w_j R(\boldsymbol{x}_i - \boldsymbol{x}_j) \\
&\qquad = \int_{\mathbb{R}^d} \sum_{i=1}^{L}\sum_{j=1}^{L} w_i w_j \cos(\boldsymbol{x}_i^\top \boldsymbol{w} - \boldsymbol{x}_j^\top \boldsymbol{w})\, f(\boldsymbol{w})\, d\boldsymbol{w} \\
&\qquad = \int_{\mathbb{R}^d} \sum_{i=1}^{L}\sum_{j=1}^{L} w_i w_j \Big\{\cos(\boldsymbol{x}_i^\top \boldsymbol{w}) \cos(\boldsymbol{x}_j^\top \boldsymbol{w}) \\
&\qquad\quad + \sin(\boldsymbol{x}_i^\top \boldsymbol{w}) \sin(\boldsymbol{x}_j^\top \boldsymbol{w})\Big\}\, f(\boldsymbol{w})\, d\boldsymbol{w} \\
&\qquad = \int_{\mathbb{R}^d} \left\{\left(\sum_{i=1}^{L} w_i \cos(\boldsymbol{x}_i^\top \boldsymbol{w})\right)^2 + \left(\sum_{i=1}^{L} w_i \sin(\boldsymbol{x}_i^\top \boldsymbol{w})\right)^2\right\} f(\boldsymbol{w})\, d\boldsymbol{w} \\
&\qquad \geq 0\,.
\end{aligned}$$

In sum, one method of constructing correlation functions is to choose a symmetric density $f(\boldsymbol{w})$ on $\mathbb{R}^d$ and evaluate $R(\boldsymbol{h})$ using (2.2.6). A valid covariance function with specified process variance $\sigma_Z^2 > 0$ and spectral density $f(\boldsymbol{w})$ is obtained from $R(\boldsymbol{h})$ by

$$C(\boldsymbol{h}) = \sigma_Z^2\, R(\boldsymbol{h})\,.$$

Example 2.2. Arguably the simplest application of (2.2.6) is when $d = 1$ and $f(w)$ is the uniform density over a symmetric interval which is taken to be $(-1/\xi, +1/\xi)$ for a given $\xi > 0$. Thus the spectral density is

$$f(w) = \begin{cases} \xi/2, & -1/\xi < w < 1/\xi \\ 0, & \text{otherwise} \end{cases}.$$

The corresponding correlation function is

$$R(h \mid \xi) = \int_{-1/\xi}^{+1/\xi} \frac{\xi}{2} \cos(hw)\, dw = \begin{cases} \dfrac{\sin(h/\xi)}{h/\xi}, & h \neq 0 \\ 1, & h = 0 \end{cases},$$

which has scale parameter ξ. Figure 2.5, which plots $R(h \mid \xi = 1/4\pi)$ over $[-1, +1]$, shows that this correlation can be used to describe processes that have both positive and negative correlations. ♦

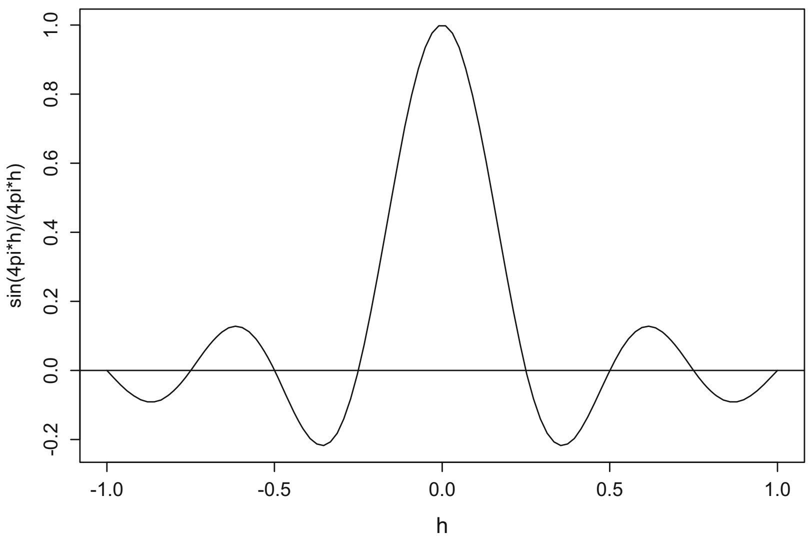


Fig. 2.5 The correlation function $R(h\,|\,\xi) = \sin(h/\xi)/(h/\xi)$ over the interval $[-1, +1]$ for $\xi = 1/4\pi$

There are additional tools that are useful for "building" covariance (correlation) functions *given* a collection of known covariance (correlation) functions. Suppose that $C_1(\cdot)$ and $C_2(\cdot)$ are valid covariance functions. Then their sum and product,

$$C_1(\cdot) + C_2(\cdot) \;\text{ and }\; C_1(\cdot) \times C_2(\cdot),$$

are also valid covariance functions. Intuitively, $C_1(\cdot) + C_2(\cdot)$ is the covariance function of the sum of two independent processes, one with covariance function $C_1(\cdot)$ and the other with covariance function $C_2(\cdot)$. Similarly, $C_1(\cdot) \times C_2(\cdot)$ is the covariance function of the product of two independent zero-mean GPs with covariances $C_1(\cdot)$ and $C_2(\cdot)$, respectively.

The product of two valid correlation functions, $R_1(\cdot)$ and $R_2(\cdot)$, is a valid correlation function, but their sum is *not* (notice that $R_1(\mathbf{0}) + R_2(\mathbf{0}) = 2$, which is not possible for a correlation function). Note, however, a convex combination of two valid correlation functions is a valid correlation function. Correlation functions that are the products of one-dimensional marginal correlation functions are sometimes called *separable* correlation functions (not to be confused with the discussion of process separability in Sect. 2.2.1).

This subsection concludes with the introduction of several families of correlation functions that have been used in the literature to specify stationary Gaussian stochastic processes (see also Journel and Huijbregts (1978), Mitchell et al. (1990), Cressie (1993), Vecchia (1988), and Stein (1999)).

Example 2.3 (Gaussian Correlation Function). The normal (Gaussian) density is a familiar symmetric density that can be used as a spectral density. To provide a simple form for the resulting $R(h)$, take the spectral density to be $N(0, 2\xi)$, where $\xi > 0$ is given. Calculation gives

$$R(h \mid \xi) = \int_{-\infty}^{+\infty} \cos(hw) \frac{1}{\sqrt{2\pi}\,\sqrt{2\xi}} \exp\left\{-w^2/(4\xi)\right\} dw$$
$$= \exp\left\{-\xi\, h^2\right\}, \quad h \in \mathbb{R}, \tag{2.2.7}$$

which has rate parameter ξ. Because of its form, $R(h \mid \xi)$ is usually called the *Gaussian correlation function.* The (separable) Gaussian family

$$R(\boldsymbol{h} \mid \boldsymbol{\xi}) = \exp\left\{-\sum_{j=1}^{d} \xi_j\, h_j^2\right\}, \quad \boldsymbol{h} \in \mathbb{R}^d, \tag{2.2.8}$$

is a legitimate correlation function because it is a product of valid correlation functions. *Separable Gaussian correlations are, far and away, the most popular family of correlation models in the computer experiments literature.*

Before describing the next parametric family of correlation functions, the reader should note that the literature considers at least four equivalent parameterizations of the Gaussian correlation function (see Sacks et al. (1989b), Higdon et al. (2008), and MacDonald et al. (2015)). These are

$$\exp\left\{-\,(h/\theta)^2\right\} = \rho^{h^2} = \rho_\star^{Ch^2} = \exp\left\{-10^{\tau}\, h^2\right\} \tag{2.2.9}$$

where $C > 0$ is a given constant (often $C = 4$ in examples) with (valid) values of the parameters being $\theta > 0$, $\rho \in (0, 1)$, $\rho_\star \in (0, 1)$, and $\tau \in (-\infty, +\infty)$. In words, θ is the scale parameter version of (2.2.7), ρ is the correlation between two inputs for which $|h| = 1$, and $\rho_\star$ is the the correlation between two inputs for which $|h| = 1/\sqrt{C}$ (e.g., when $C = 4$ and $d = 1$, $\rho_\star = Cor[Y(0), Y(1/2)]$). Using simple algebra, the parameter defining any of these correlations can be expressed in terms of any of the other four parameters. The reasons that authors select a particular parameterization are for their ease of interpretation, importance for prior specifications of Bayesian methodology, and for their ability to enhance optimization of the likelihood and related functions; Sect. 3.6.3 will discuss these issues further. ♦

Example 2.4 (Power Exponential Family). The Gaussian correlation functions are special cases of *power exponential correlations*. The function

$$R(h \mid \xi) = \exp\{-\xi\, |h|^p\}, \quad h \in \mathbb{R}, \tag{2.2.10}$$

is said to be a power exponential correlation function provided $\xi > 0$ and $0 < p \leq 2$. In addition to the Gaussian ($p = 2$) subfamily, the GP having correlation function (2.2.10) with $p = 1$,

$$R(h \mid \xi) = \exp\{-\xi\, |h|\},$$

is known as the Ornstein–Uhlenbeck process.

For later reference, note that every power exponential correlation function, $0 < p \leq 2$, is continuous at the origin, and only the Gaussian correlation function is differentiable at the origin. In fact, the Gaussian correlation function is infinitely differentiable at the origin. The d-dimensional separable version of the power exponential correlation,

$$R(\boldsymbol{h} \mid \boldsymbol{\xi}) = \exp\left\{ -\sum_{j=1}^{d} \xi_j \, |h_j|^{p_j} \right\}, \quad \boldsymbol{h} \in \mathbb{R}^d, \tag{2.2.11}$$

is also a valid correlation function. ♦

Example 2.5 (Matérn Correlation Family). In his thesis, Matérn (1960) introduced the correlation function that bears his name (see the Matérn (1986) reprint and Vecchia (1988) for related work). The Matérn correlation model has been used widely to model environmental data (see Rodríguez-Iturbe and Mejía (1974); Handcock and Stein (1993); Handcock and Wallis (1994) and especially Stein (1999)).

From the viewpoint of Bochner's formula (2.2.6), the Matérn correlation function arises by choosing the t distribution with parameters $\nu > 0$ and $\psi > 0$,

$$f(w) = \frac{\Gamma(\nu + 1/2)}{\Gamma(\nu)\sqrt{\pi}} \left(\frac{4\nu}{\psi^2} \right)^{\nu} \frac{1}{(w^2 + 4\nu/\psi^2)^{\nu+1/2}}, \quad w \in \mathbb{R},$$

as spectral density. The result is the two parameter correlation family

$$R(h \mid (\nu, \psi)) = \frac{1}{\Gamma(\nu) 2^{\nu-1}} \left(\frac{2\sqrt{\nu}\,|h|}{\psi} \right)^{\nu} K_{\nu}\left(\frac{2\sqrt{\nu}\,|h|}{\psi} \right),$$

where $K_\nu(\cdot)$ is the modified Bessel function of order ν and ψ is a scale parameter. The modified Bessel function arises as the solution of a certain class of ordinary differential equations (Kreyszig (1999)). In general, $K_\nu(t)$ is expressed in terms of an infinite power series in t although it can be written in a simple form for some ν.

When $\nu = 1/2$,

$$K_{1/2}(t) = \sqrt{\pi}\, e^{-t} / \sqrt{2t} \;\text{ with }\; R(h \mid (1/2, \psi)) = e^{-\sqrt{2}|h|/\psi},$$

which is a special case of the power exponential correlation function with $p = 1$ that was introduced earlier. Similarly, $R(h \mid (\nu, \psi)) \to e^{-(h/\psi)^2}$ as $\nu \to \infty$ so that the Matérn correlation family includes the Gaussian correlation function in the limit.

A valid Matérn correlation for d-dimensional inputs is formed from any product of one-dimensional Matérn correlation functions. For example, the family

$$R(\boldsymbol{h} \mid (\nu, \boldsymbol{\psi})) = \prod_{i=1}^{d} \frac{1}{\Gamma(\nu) 2^{\nu-1}} \left(\frac{2\sqrt{\nu}\,|h_i|}{\psi_i} \right)^{\nu} K_{\nu}\left(\frac{2\sqrt{\nu}\,|h_i|}{\psi_i} \right)$$

uses dimension-specific scale parameters $\psi_1, \ldots, \psi_d$ and a common smoothness parameter ν. ♦

Example 2.6 (Cubic Correlation Family). Cubic correlation functions have the form

$$R(h \mid \psi) = \begin{cases} 1 - 6\left(\frac{h}{\psi}\right)^2 + 6\left(\frac{|h|}{\psi}\right)^3, & |h| \le \psi/2 \\ 2\left(1 - \frac{|h|}{\psi}\right)^3, & \psi/2 < |h| \le \psi \\ 0, & |h| > \psi \end{cases}, \tag{2.2.12}$$

for $h \in \mathbb{R}$, where $\psi > 0$. The spectral density that produces (2.2.12) is proportional to

$$\frac{1}{w^4\psi^3}\left\{72 - 96\cos(w\psi/2) + 24\cos(w\psi)\right\}.$$

In addition to being a piecewise cubic polynomial, $R(h \mid \psi)$ has continuous derivatives at all $h \in \mathbb{R}$; the right column of Fig. 2.8 shows the smoothness of $R(h \mid \psi)$ when $\psi \in \{0.5, 1.0, 10.0\}$. The cubic correlation function has compact support because $R(h \mid \psi) = 0$ whenever inputs x_1 and x_2 are sufficiently far apart, i.e., whenever $|x_1 - x_2| > \psi$. Numerically, this property means that the correlation matrix of $Z(x)$ can have substantial numbers of zero entries which allows sparse matrix methods to be used for operations such as matrix inversions. Anticipating Sect. 3.2 on best linear unbiased prediction in computer experiments, using (2.2.12), leads to cubic spline interpolating predictors.

As with previous examples,

$$R(\boldsymbol{h} \mid \boldsymbol{\psi}) = \prod_{j=1}^{d} R(h_j \mid \psi_j), \quad \boldsymbol{h} \in \mathbb{R}^d, \tag{2.2.13}$$

is a multiple input separable generalization of the cubic correlation function. The form (2.2.13) permits dimension-specific distances at which $Z(\cdot)$ values are uncorrelated. Other one-dimensional cubic correlation functions can be found in Mitchell et al. (1990) and Currin et al. (1991). ♦

Example 2.7 (Bohman Correlation Family). In addition to the cubic correlation function, other compactly supported correlation functions have been proposed in the literature because of their computational tractability (see Kaufman et al. (2011)). One important such function is the *Bohman correlation function*

$$R(h \mid \psi) = \begin{cases} \left(1 - \frac{|h|}{\psi}\right)\cos\left(\frac{\pi|h|}{\psi}\right) + \frac{1}{\pi}\sin\left(\frac{\pi|h|}{\psi}\right), & |h| < \psi \\ 0, & \psi \le |h| \end{cases} \tag{2.2.14}$$

for $h \in \mathbb{R}$, where $\psi > 0$ is a fixed scale parameter (see also Gneiting (2002); Stein (2008)). ♦

2.2.3 Using the Correlation Function to Specify a GP with Given Smoothness Properties

This subsection discusses smoothness properties of realizations of

$$Y(\boldsymbol{x}) = \boldsymbol{f}^\top(\boldsymbol{x})\boldsymbol{\beta} + Z(\boldsymbol{x})$$

in (2.2.3). This discussion can be separated into the analysis of the smoothness of the regression term, $\boldsymbol{f}^\top(\boldsymbol{x})\boldsymbol{\beta}$, and the smoothness of draws from the stationary GP, $Z(\cdot)$. Determining the continuity and differentiability of the regression terms is straightforward, requiring standard calculus.

The remainder of this section will consider properties of realizations from $Z(\cdot)$ which will be related to the covariance function of $Z(\cdot)$, say $C(\cdot)$. Because of its technical ease, this treatment is initiated by considering *mean square (MS) continuity*. Mean square properties describe the average performance over the sample paths rather than the properties of individual sample paths, recognizing that the latter are of greater interest when modeling computer simulator output.

Definition. Suppose $W(\cdot)$ is a stationary process with input domain $\mathcal{X}$ that has a positive, finite second moment; $W(\cdot)$ is said to be *MS continuous* at $\boldsymbol{x}_0 \in \mathcal{X}$ provided

$$\lim_{\boldsymbol{x}\to\boldsymbol{x}_0} E\left[(W(\boldsymbol{x}) - W(\boldsymbol{x}_0))^2\right] = 0\,.$$

The process is *MS continuous on* $\mathcal{X}$ provided it is MS continuous at every $\boldsymbol{x}_0 \in \mathcal{X}$.

Suppose $C(\cdot)$ is the covariance function of the stationary process $Z(\cdot)$, then

$$E\left[(Z(\boldsymbol{x}) - Z(\boldsymbol{x}_0))^2\right] = 2\left(C\left(\boldsymbol{0}\right) - C\left(\boldsymbol{x} - \boldsymbol{x}_0\right)\right). \tag{2.2.15}$$

The right-hand side of (2.2.15) shows that $Z(\cdot)$ is MS continuous at $\boldsymbol{x}_0$ provided $C(\boldsymbol{h})$ is continuous at the origin—in fact, $Z(\boldsymbol{x})$ is MS continuous at *every* $\boldsymbol{x}_0 \in \mathcal{X}$ provided $C(\boldsymbol{h})$ is continuous at the origin.

Assume $Z(\boldsymbol{x})$ has positive variance so that its correlation function satisfies $C(\boldsymbol{0}) = \sigma_Z^2 > 0$. Then the continuity condition $C(\boldsymbol{h}) \to C(\boldsymbol{0}) = \sigma_Z^2 > 0$ as $\boldsymbol{h} \to \boldsymbol{0}$ is equivalent to requiring

$$R(\boldsymbol{h}) = C(\boldsymbol{h})/\sigma_Z^2 \to 1.0 \text{ as } \boldsymbol{h} \to \boldsymbol{0} \tag{2.2.16}$$

for the correlation function. All the correlation functions listed in Sect. 2.2.2 are continuous at the origin.

Now consider the primary objective of this section that of identifying conditions on the process that guarantee (with probability one) a target, "smoothness" property holds for realizations $z(\boldsymbol{x})$ from a $Z(\boldsymbol{x})$ process. Let "Q" denote the desired property. The goal is that

$$P\left[\omega : Z(\cdot, \omega) \text{ has property } Q\right] = 1\,.$$

Such behavior is termed *almost sure (a.s.) behavior* of the sample draws.

As the first example of such a condition, Adler (1981) (page 60) shows that the sample draws of stationary GPs will be almost surely continuous provided a small requirement is added to the continuity condition (2.2.16) that guarantees MS continuity. Stated in words, a.s. continuity holds for the draws from a stationary GP provided $R(\boldsymbol{h})$ converges to unity *sufficiently fast* as $\boldsymbol{h} \to \boldsymbol{0}$, i.e., for some $c > 0$, $\epsilon > 0$, and $\delta < 1$,

$$1 - R(\boldsymbol{h}) \leq \frac{c}{|\ell n\,(\|\boldsymbol{h}\|_2)\,|^{1+\epsilon}} \text{ for all } \|\boldsymbol{h}\|_2 < \delta\,, \tag{2.2.17}$$

where $\|\cdot\|_2$ denotes Euclidean distance. Equation (2.2.17) is a stronger version of $R(\boldsymbol{h}) \to 1$ because the factor $|\,\ell n\,(\|\boldsymbol{h}\|_2)\,|^{1+\epsilon} \to +\infty$ as $\boldsymbol{h} \to 0$.

As another example, consider the a.s. differentiability of draws from a stationary process $Z(\boldsymbol{x})$; a draw is a function $Z(\boldsymbol{x}, \omega)$, $\boldsymbol{x} \in \mathbb{R}^d$, corresponding to $\omega \in \Omega$. Suppose that the j^{th} partial derivative of $Z(\boldsymbol{x}, \omega)$ exists at $\boldsymbol{x}_0 \in \mathcal{X}$ for $j = 1, \ldots d$, i.e.,

$$\nabla_j Z(\boldsymbol{x}_0, \omega) = \lim_{\delta \to 0} \frac{Z(\boldsymbol{x}_0 + \boldsymbol{e}_j\,\delta, \omega) - Z(\boldsymbol{x}_0, \omega)}{\delta}$$

exists where $\boldsymbol{e}_j$ denotes the unit vector in the j^{th} direction. Below, conditions are stated for the covariance function that guarantee that the sample paths are a. s. differentiable. As motivation for this condition, observe that the following heuristic calculation gives the covariance of the slope of the secant of the $Z(\boldsymbol{x})$ process at inputs $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ in $\mathcal{X}$. When the second partial derivative of $C(\cdot)$ exists,

$$\begin{aligned}
&Cov\left[\frac{1}{\delta_1}\left(Z(\boldsymbol{x}_1 + \boldsymbol{e}_j\,\delta_1) - Z(\boldsymbol{x}_1)\right), \frac{1}{\delta_2}\left(Z(\boldsymbol{x}_2 + \boldsymbol{e}_j\,\delta_2) - Z(\boldsymbol{x}_2)\right)\right] \\
&\qquad = \frac{1}{\delta_1\delta_2}\Big\{C(\boldsymbol{x}_1 - \boldsymbol{x}_2 + \boldsymbol{e}_j\,(\delta_1 - \delta_2)) - C(\boldsymbol{x}_1 - \boldsymbol{x}_2 + \boldsymbol{e}_j\,\delta_1) \\
&\qquad\quad - C(\boldsymbol{x}_1 - \boldsymbol{x}_2 - \boldsymbol{e}_j\,\delta_2) + C(\boldsymbol{x}_1 - \boldsymbol{x}_2)\Big\} \\
&\qquad \to -\left.\frac{\partial^2 C(\boldsymbol{h})}{\partial h_j^2}\right|_{\boldsymbol{h}=\boldsymbol{x}_1-\boldsymbol{x}_2} \quad \text{as } \delta_1, \delta_2 \to 0.
\end{aligned}$$

This calculation suggests that the partial derivatives of $Z(\cdot)$ and the covariance function $C(\boldsymbol{h})$ are linked. Indeed, the stationary GP $Z(\boldsymbol{x})$ has a.s. j^{th} partial derivative, $j = 1, \ldots, d$, provided

$$C_j^{(2)}(\boldsymbol{h}) \equiv \frac{\partial^2 C(\boldsymbol{h})}{\partial h_j^2}$$

exists and is continuous with $C_j^{(2)}(\boldsymbol{0}) \neq 0$ and $R_j^{(2)}(\boldsymbol{h}) \equiv C_j^{(2)}(\boldsymbol{h})/C_j^{(2)}(\boldsymbol{0})$ satisfies (2.2.17). In this case $-C_j^{(2)}(\boldsymbol{h})$ is the covariance function of the process $\nabla_j Z(\boldsymbol{x})$ and $R_j^{(2)}(\boldsymbol{h})$ is its correlation function. Conditions that ensure higher-order $Z(\cdot)$ derivatives exist can be iteratively developed in the same way.

This section concludes by illustrating the effects of changing the parameters in two of the correlation families introduced earlier. These effects will be shown in draws from a *zero mean*, *unit variance* stationary GP over $[0, 1]$ with the specified correlation family. Of course, adding regression terms $\boldsymbol{f}^\top(\boldsymbol{x})\boldsymbol{\beta}$ would further provide flexibility in the types of functions $y(\boldsymbol{x})$ that can be modeled, however the use of regression terms will not be illustrated here.

Example 2.4 (Continued–Power Exponential Correlation). Figures 2.6 and 2.7 show $z(x)$ draws from the power exponential correlation function (2.2.10). Figure 2.6 fixes the rate parameter $\xi = 1.0$; it illustrates the effect of varying the power, p, on the $z(x)$ draws. Figure 2.7 fixes $p = 2$; it shows the effect of varying ξ on the $z(x)$ draws.

The bottom two rows of Fig. 2.6 have powers $p < 2$; it has been previously stated that sample paths $z(x) = Z(x, \omega)$ for $p < 2$ are *not differentiable* which explains the "wiggly" behavior of these $z(x)$. The $z(x)$ draws in the top row correspond to $p = 2.0$; they are infinitely differentiable. Indeed, the $z(x)$ draws in the top row of Fig. 2.6 are very near the process mean of zero.

Figure 2.7 shows that the number of local maxima and minima of $z(x)$ is controlled by the rate parameter ξ when $p = 2.0$. As ξ *increases*, the correlations between each fixed pair of inputs *decreases* and the number of local maxima of $z(x)$ *increases*. Indeed, the process $Z(x)$ "wiggles" more like white noise, as $\xi \to +\infty$. The most extreme version of this phenomenon shown in Fig. 2.7 is the bottom row where $\xi = 100.0$. ♦

Example 2.6 (Continued—Cubic Correlation). Recall that the cubic correlation (and covariance) function (2.2.12) is twice continuously differentiable. Thus $z(x)$ draws from a GP with cubic correlation will be continuous and differentiable. Figure 2.8 shows draws from this GP for different ψ. As the scale parameter ψ *decreases*, the domain where $R(h) = 0$ *increases* and hence the paths become more like white noise, i.e., having independent and identically distributed Gaussian components. As ψ *increases*, $R(h)$ becomes near 1.0 for larger h ranges (see top row) meaning that $z(x)$ will also be more nearly flat over larger x ranges as its values move nearly in lock step. ♦

2.3 Increasing the Flexibility of the GP Model

This section provides an overview of some of the strategies that have been suggested in the literature for replacing the regression + stationary GP model:

$$Y(\boldsymbol{x}) = \boldsymbol{f}^\top(\boldsymbol{x})\boldsymbol{\beta} + Z(\boldsymbol{x}) \tag{2.3.1}$$

with more flexible $Y(\boldsymbol{x})$ to produce broader classes of $y(\boldsymbol{x})$ (see (2.2.3) for $Y(\boldsymbol{x})$ details). Consider Fig. 2.9 that illustrates two functions $y(\boldsymbol{x})$ that would not be well

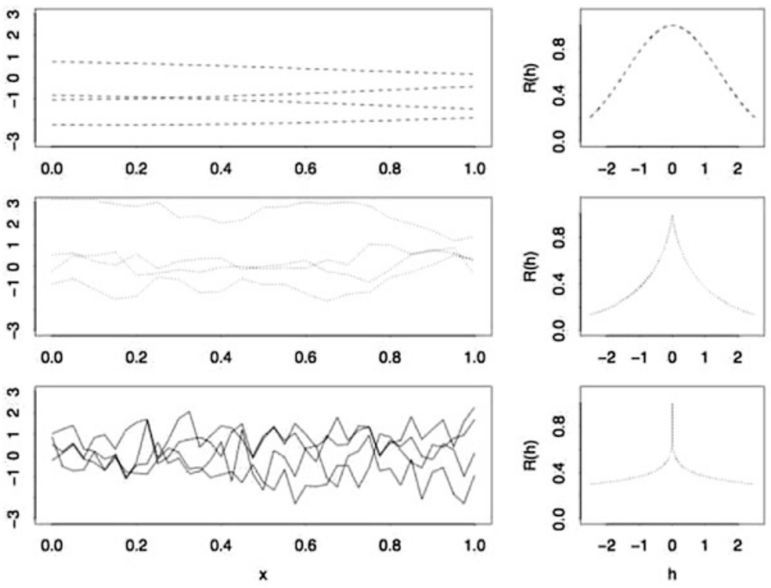


Fig. 2.6 Left column: the effect of varying the power p in the power exponential correlation function (2.2.10) on sets of four draws from a zero mean, unit variance GP with fixed $\xi \equiv 1.0$; right column: $C(h) = \exp(-|h|^p)$, $-2.0 \le h \le +2.0$. Here $p = 2.0$ (top row), $p = 0.75$ (middle row), and $p = 0.20$ (bottom row)

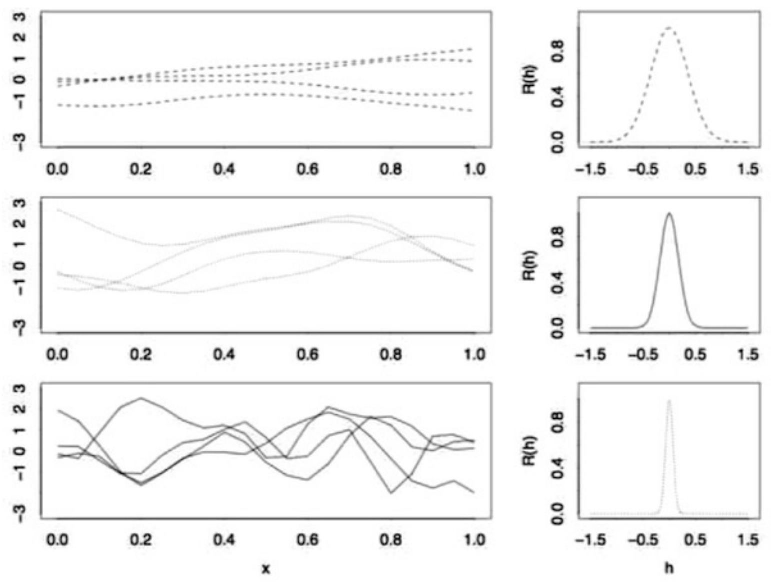


Fig. 2.7 Left column: the effect of varying the rate parameter ξ in the power exponential correlation function (2.2.10) on sets of four draws from a zero mean, unit variance GP with fixed $p \equiv 2.0$; right column: $C(h) = \exp(-\xi h^2)$, $-1.5 \le h \le +1.5$. Here $\xi = 4$ (top row), $\xi = 16$ (middle row), and $\xi = 100$ (bottom row)

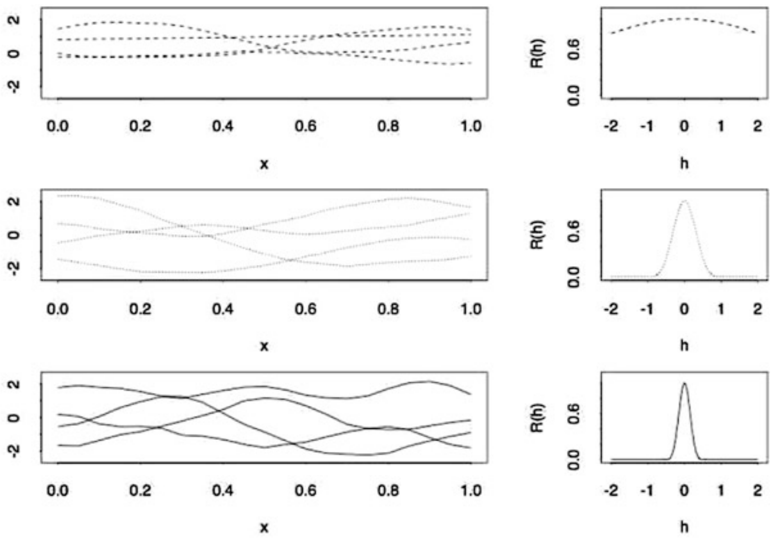


Fig. 2.8 Left column: the effect of varying the scale parameter ψ in the cubic correlation function (2.2.12) on sets of four draws from a zero mean, unit variance GP; right column: cubic correlation $R(h \mid \psi)$, $-2.0 \le h \le +2.0$. Here $\psi = 10.0$ (top row), $\psi = 1.0$ (middle row), and $\psi = 0.5$ (bottom row)

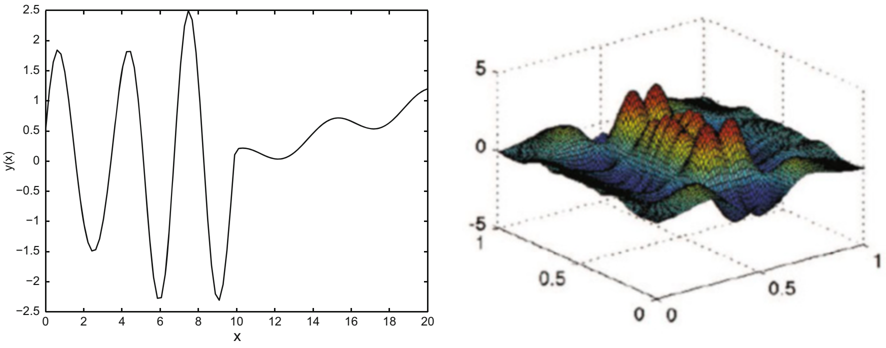


Fig. 2.9 Left panel: a nonstationary function $y(x)$ defined on $[0, 20]$; right panel: a nonstationary function $y(x_1, x_2)$ defined on $[0, 1]^2$

modeled by a $Y(\boldsymbol{x})$ of the form (2.3.1). The left-hand panel plots a $y(x)$ with $x \in [0, 20]$ for which the range of the function and its global trend on $[0, 10]$ are visibly different than on $[10, 20]$. Similarly in the right-hand panel of Fig. 2.9, the number of local maxima and minima of $y(x_1, x_2)$ is quite different in the "central" rectangular portion of the input space than in its complement. Both $\boldsymbol{y}(\boldsymbol{x})$ show nonstationarity. In practice, many environmental studies require nonstationary models as well as other subject-matter work that involves phase changes or multiple boundary conditions (Cressie and Wikle (2011); Bornn et al. (2012)).

One method of producing nonstationary $Y(\boldsymbol{x})$ is to convolve a stationary process with a kernel. For example, Higdon et al. (1999) integrate white noise, the spatial analog of a random sample of normal observations, against a Gaussian kernel to produce a nonstationary $Y(\boldsymbol{x})$. In the same spirit, Haas (1995) constructs $Y(\boldsymbol{x})$ models as a moving window over a stationary process. Sampson and Guttorp (1992) introduced another approach to producing nonstationary $Y(\boldsymbol{x})$ that begins with a stationary process; they deformed the input $\boldsymbol{x}$ of a stationary process to form $Y(\boldsymbol{x})$ (see also Guttorp and Sampson (1994) and Guttorp et al. (1994)). Bornn et al. (2012) adopt a related methodology to construct nonstationary $Y(\boldsymbol{x})$ which regards $Y(\boldsymbol{x})$ as the projection of a higher-dimensional stationary process $Y^{\star}(\boldsymbol{x}^{\star})$; they seek to identify the additional dimensions to recover the stationarity process. Standard dimension reduction techniques can be employed to analyze the resulting expanded model.

The remainder of this section will present two additional strategies for forming flexible $Y(\boldsymbol{x})$ models. These ideas will be employed in later sections.

2.3.1 Hierarchical GP Models

This subsection introduces *hierarchical* GP models. In addition as a tool for producing a flexible $Y(\boldsymbol{x})$, hierarchical models represent a fundamental paradigm shift because they add information to the statement of a process model—hierarchical models specify distributions to describe the likely model parameter values.

Example 2.8. This example illustrates the variety of $y(\boldsymbol{x})$ that can be produced by a simple hierarchical model. Suppose that output $y(x)$, $0 \le x \le 1$, can be described as a draw from

$$Y(x) = \beta_0 + Z(x), \quad 0 \le x \le 1\,, \tag{2.3.2}$$

where $Z(\cdot)$ is a stationary Gaussian random field with zero mean, precision (inverse variance) λ_z, and Gaussian correlation function with parameterization

$$R(h\,|\,\rho) = \rho^{h^2}$$

(see Eqs. (2.2.7) and (2.2.9)). The location of the intercept is determined by β_0, the range of the $y(x)$ values is determined by λ_z, the process precision, and the number of local maxima and minima is determined by ρ. Rather than specifying a single $(\beta_0, \lambda_z, \rho)$, the hierarchical model takes draws of the model parameters from a distribution that embodies knowledge about their values. For example, suppose that it is assumed that the distributions for each parameter are mutually independent. Further assume it is known that β_0 is likely to be near 20 and the certainty in this knowledge is described by $\beta_0 \sim N(20, 4^2)$, while σ_z is likely to be near 1.5 with prior knowledge described by $\lambda_z \equiv 1/\sigma_z^2 \sim \Gamma(6.25, 25)$, while $\rho \sim Be(2, 3)$. Five draws from this prior distribution produce the values listed in Table 2.1. Taking the parameters from each row of Table 2.1 and forming a realization from (2.3.2) give the five $y(x)$ draws shown in Fig. 2.10. Notice the greater variety of $y(x)$ draws that are possible from the hierarchical model compared with those from fixed parameter GP models; in contrast each panel of Figs. 2.6, 2.7, and 2.8 plotted five draws from a fixed parameter GP. ♦

Prior draw	$\beta_0 \sim N(20, 4^2)$	$\lambda_z \sim \Gamma(6.25, 25)$	σ_z	$\rho \sim Be(2, 3)$
1	15.73	0.43	1.52	0.50
2	16.76	0.35	1.68	0.33
3	8.22	0.21	2.19	0.01
4	25.75	0.17	2.44	0.12
5	21.30	0.14	2.66	0.81

Table 2.1 Five draws from informative $[\beta_0, \lambda_z, \rho]$ prior having mutually independent components

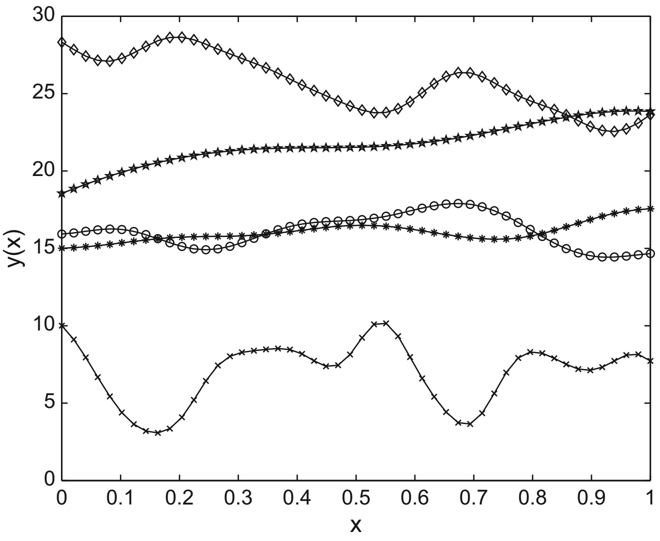


Fig. 2.10 Five draws from (2.3.2), one corresponding to each of the rows in Table 2.1

Section 4.3 develops a fully Bayesian predictor based on a hierarchical model whose top stage, given parameters $(\boldsymbol{\beta}, \lambda_z, \boldsymbol{\rho})$, is

$$Y(\boldsymbol{x}) = \sum_{j=1}^{p} f_j(\boldsymbol{x})\beta_j + Z(\boldsymbol{x}) = \boldsymbol{f}^\top(\boldsymbol{x})\boldsymbol{\beta} + Z(\boldsymbol{x}), \tag{2.3.3}$$

where $Z(\cdot)$ is defined following (2.3.2). As in Example 2.8, the second-stage prior distribution, $[\boldsymbol{\beta}, \lambda_z, \boldsymbol{\rho}]$, is specified in "pieces." Suppose that it is reasonable to assume that the regression effect parameters $\boldsymbol{\beta}$, which specify *global trends*, and the precision λ_Z, which determines the magnitude of *local deviations*, are independent of the correlation parameters $\boldsymbol{\rho}$, which describe the number of local maxima and minima. This means that

$$[\boldsymbol{\beta}, \lambda_Z, \boldsymbol{\rho}] = [\boldsymbol{\beta}, \lambda_Z] \times [\boldsymbol{\rho}] = [\boldsymbol{\beta} \mid \lambda_Z] \times [\lambda_Z] \times [\boldsymbol{\rho}] \,,$$

where the second equality holds because $[\boldsymbol{\beta}, \lambda_Z] = [\boldsymbol{\beta} \mid \lambda_Z] \times [\lambda_Z]$. Thus the overall prior can be determined from these three pieces.

Building a rational prior requires hard work. See Oakley (2002) for a method of prior specification and a case study. Other examples of the construction of prior distributions for parameters can be found in the environmental literature. For example, Handcock and Wallis (1994) build a prior distribution for correlation parameters in their space–time model of the mean temperature of a region of the Northern USA.

The previous paragraph describes what might be thought of as an *informative* $[\boldsymbol{\beta}, \lambda_Z]$ prior. In some applications, there may not be adequate subject matter knowledge of the output to specify an informative prior. In cases where informative priors cannot be readily formed, it is tempting to use so-called *non-informative* priors that give "equal" weight to all the legitimate parameter values. The reader should be warned that there is not always agreement in the statistical community about what constitutes a non-informative prior, even for parameters having finite ranges. One reason is that a prior based on an equal weighting of the parameter values on one scale does not (ordinarily) correspond to an equal weighing of a 1-1 transform of that parameter. For example, choosing $\rho \in [0, 1]$ to have a uniform distribution over $[0, 1]$ is not the same as the prior choice $\theta \propto 1$ where $\rho = \exp(-\theta)$. In addition, not every choice of a non-informative prior pairs with the first-stage model (2.3.3) to produce a legitimate posterior for $y(\cdot)$ (see Berger et al. (2001)). More will be said about non-informative second-stage priors in Sects. 3.3.5 and 4.3 which discuss "posterior mode empirical best linear unbiased predictors."

A third possible choice for a $[\boldsymbol{\beta}, \lambda_Z, \boldsymbol{\rho}]$ prior is a "conjugate" prior. Conjugate priors have the property that their posterior distributions come from the same parametric family. Section 4.2 provides examples of conjugate second-stage distributions.

2.3.2 Other Nonstationary Models

Another method of enhancing (2.3.1) is to replace the regression term $\boldsymbol{f}^\top(\boldsymbol{x})\boldsymbol{\beta}$ by a more flexible global trend model. The $\boldsymbol{f}^\top(\boldsymbol{x})\boldsymbol{\beta}$ model assumes that the researcher knows the regression variables to be used in $Y(\boldsymbol{x})$, i.e., knows additive functions $\boldsymbol{f}(\boldsymbol{x})$ that provide the global behavior of $y(\boldsymbol{x})$ aside from regression parameters.

For simplicity suppose that output $y(\boldsymbol{x})$ is defined for $\boldsymbol{x}$ in a hyper-rectangle $\mathcal{X} = \bigtimes_{\ell=1}^{d} [a_\ell, b_\ell]$. In their proposal of calibration methodology, Chakraborty et al. (2013) suggest using the nonparametric MARS regression function in place of $\boldsymbol{f}^\top(\boldsymbol{x})\boldsymbol{\beta}$ (see Chap. 8). Arguably, a more straightforward $Y(\boldsymbol{x})$ that allows the type of behavior illustrated in Fig. 2.9 was introduced by Gramacy and Lee (2008); these authors used *treed Gaussian processes* (TGPs) to provide models that can represent nonstationary functions. The TGP model assumes that $\mathcal{X}$ can be partitioned into hyper-rectangles so that within each partition, a partition-specific regression + stationary model provides a reasonable description of $y(\boldsymbol{x})$. For example, in the left-hand panel of Fig. 2.9, the portions of $y(x)$ on $[0, 10]$ and on $[10, 20]$ each appear to be reasonably described by their own stationary processes. Similarly, the "central" rectangular portion of the right-hand panel of Fig. 2.9 shows one type of behavior, while the surrounding rectangles exhibit other types of stationary behavior.

Gramacy and Lee (2008) use the Bayesian tree methodology of Chipman et al. (1998) to partition $\mathcal{X}$. They extend the work of Chipman et al. (2002) to allow linear trends independently within each of the regions. As always, the methodology for prediction using such a model requires adequate data so that the separate stationary models can be fit for each input partition (see Loeppky et al. (2009)).

2.4 Models for Output Having Mixed Qualitative and Quantitative Inputs

Suppose the simulator produces deterministic output $y(\boldsymbol{x}, t)$ that depends on the quantitative input variables $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)^\top$ and a qualitative variable having T levels, here indexed by t. One can interpret $y(\boldsymbol{x}, t)$ as determining t different curves or response surfaces, indexed by t.

If, in fact, the output depends on $Q > 1$ qualitative variables, with the q^{th} qualitative variable having T_q levels, assume that the $T = \prod_{q=1}^{Q} T_q$ possible combinations of levels are indexed by a single symbol taking on values from 1 to T (lexicographically ordered). This suppresses the inherent factorial structure, and this issue will be considered later.

Model $y(\boldsymbol{x}, t)$ as a draw from

$$Y(\boldsymbol{x}, t) = \sum_{j=1}^{p} f_j(\boldsymbol{x}, t)\beta_j + Z_t(\boldsymbol{x}) \tag{2.4.1}$$

where all terms are as in (2.2.3) and $Z_t(\boldsymbol{x})$ is a mean zero, stationary GP with variance $\sigma^2_{Z_t}$. If each value of t is viewed as determining a separate response surface, one can fit separate GP models to each of these response surfaces. However, if the response surfaces are similar, perhaps it is possible to do better by developing predictors for each surface that "borrow" information from the other surfaces. In multiple regression this is accomplished by using indicator variables to represent different response surfaces. For example, one can use indicator variables to write a single multiple regression model that represents several lines. This single model has more degrees of freedom for error than fitting separate lines, although this comes at the expense of having to assume the error variance is the same for each line.

Is it possible to mimic what is done in multiple regression and use indicator variables to model the effects of qualitative inputs in GP models? What happens if indicator variables are added to our model and treated as though they are quantitative? The use of indicator variables in the regression portion of (2.4.1) (in particular, allowing some of the $f_j(\boldsymbol{x}, t)$ to involve indicator variables) presents no difficulties. Unfortunately, using indicator variables in the correlation function in a "naíve" manner does create problems. To see this, consider the following. For $1 \le t \le T$ define

$$I_t(i) = \begin{cases} 1 \text{ if } i = t \\ 0 \text{ otherwise} \end{cases}.$$

Treating these as quantitative, the Gaussian correlation function (see (2.2.8)) becomes

$$\begin{aligned} R((\boldsymbol{x}_1, t_1), (\boldsymbol{x}_2, t_2)) &= \prod_{l=1}^{T} \exp\left\{-\xi_l^* \left(I_l(t_1) - I_l(t_2)\right)^2\right\} \times \prod_{k=1}^{d} \exp\left\{-\xi_k \left(x_{1,k} - x_{2,k}\right)^2\right\} \\ &= \exp\left\{-\xi_{t_1}^*\right\} \times \exp\left\{-\xi_{t_2}^*\right\} \times \prod_{k=1}^{d} \exp\left\{-\xi_k \left(x_{1,k} - x_{2,k}\right)^2\right\} \\ &= \tau_{t_1}\tau_{t_2} \times \prod_{k=1}^{d} \exp\left\{-\xi_k \left(x_{1,k} - x_{2,k}\right)^2\right\} \end{aligned}$$

for $t_1 \ne t_2$ where $\tau_{t_j} = \exp\left\{-\xi_{t_j}^*\right\}$. Notice $0 < \tau_{t_j} < 1$.

For an intuitive argument for why this approach is unsatisfactory, suppose $T = 4$ with response surfaces 1 and 2 very similar, response surfaces 3 and 4 very similar, but response surfaces 1 and 3 very different. In particular, suppose 1 and 2 are very similar in the sense that they are highly correlated implying that $\tau_1\tau_2$ is close to 1. In this case both τ_1 and τ_2 must be close to 1. Similarly, if 3 and 4 are highly correlated, $\tau_3\tau_4$ should be close to 1, and hence both τ_3 and τ_4 must be close to 1. However, if 1 and 3 are essentially uncorrelated, one would expect $\tau_1\tau_3$ to be close to 0. But this is impossible if both τ_1 and τ_3 are close to 1. Of course, this assumes the τ_i can be interpreted as correlations between response surfaces.

This suggests that a "naíve" use of indicator variables to represent qualitative variables (similar to what is done in standard regression) imposes undesirable constraints on the "between response surfaces" correlations. One must model the qualitative variables more carefully, at least in terms of the correlation structure.

For simplicity, in what follows a constant mean model of the form

$$Y(\boldsymbol{x}, t) = \beta_t + Z_t(\boldsymbol{x}) \tag{2.4.2}$$

is assumed.

One approach to incorporating indicator variables into the correlation function is suggested by Kennedy and O'Hagan (2000). They assume one has a collection of multi-fidelity computer simulations (simulations of differing degrees of accuracy), each involving the same quantitative factors. These simulations are modeled collectively by a single computer model with a common set of quantitative factors and a single qualitative factor to describe the different degrees of accuracy of the simulations. This approach implicitly assumes that as the level of the qualitative factor changes (increases), the fidelity increases. Thus, it may not be appropriate for the general case of incorporating a qualitative variable.

An approach that is popular in the literature is introduced in Qian et al. (2008) and Zhou et al. (2011). This approach assumes

$$Cor\left[Z_{t_1}(\boldsymbol{x}_1), Z_{t_2}(\boldsymbol{x}_2)\right] = \tau_{t_1,t_2} \prod_{i=1}^{d} \exp\left\{-\xi_i \left(x_{1,i} - x_{2,i}\right)^2\right\} \tag{2.4.3}$$

where τ_{t_1,t_2} is the "cross-correlation" between the response surfaces corresponding to "categories" t_1 and t_2 of the qualitative variable. The $T \times T$ matrix $\boldsymbol{\tau} = \{\tau_{r.s}\}$ is assumed to be a positive definite matrix with unit diagonal elements to guarantee that the matrix of correlations whose i, jth entry is $Cor[Z_{t_i}(\boldsymbol{x}_i), Z_{t_j}(\boldsymbol{x}_j)]$ is a valid correlation matrix. In addition, one assumes $\sigma^2_{Z_t} = \sigma^2_Z$ for all t.

Example 2.9. We consider the simple setting of a single qualitative variable with $T = 2$ levels and a single quantitative variable x, with $0 \le x \le 1$. Figure 2.11 shows four draws from the constant mean model (2.4.2) with correlation as in (2.4.3). We set $\beta_1 = 0$ (black curve), $\beta_2 = 1$ (red curve), $\sigma^2_Z = 1$, and $\xi = 50$. A value of $\xi = 50$ guarantees a moderate amount of "wiggliness." In (a), the cross-correlation, $\tau_{1,2}$, is

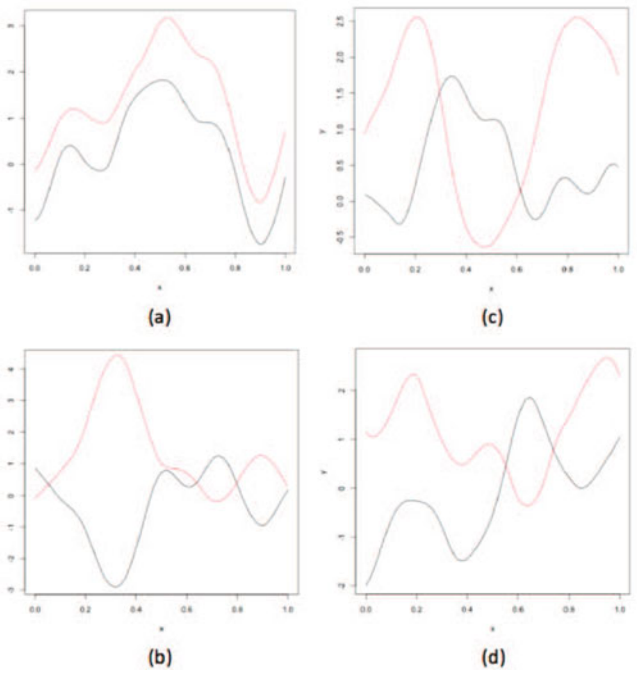


Fig. 2.11 Realizations for cross-correlation equal to (**a**) $\tau_{1,2} = 0.9$, (**b**) $\tau_{1,2} = -0.9$, (**c**) $\tau_{1,2} = 0.1$, and (**d**) $\tau_{1,2} = -0.1$

0.9, in (b) the cross-correlation is −0.9, in (c) the cross-correlation is 0.1, and in (d) the cross-correlation is −0.1. Figure 2.11 provides some insight into the model.

Each level of the qualitative variable corresponds to a curve representing a draw from a GP like those in Fig. 2.7. The term "cross-correlation" suggests that this parameter controls how correlated (similar) pairs of curves are. The figure supports this. For both (a) and (b), the cross-correlation is large. In (a) the two curves are similar. In (b) the two curves reflect what we might intuitively regard as strongly negatively correlated curves. For both (c) and (d), the cross-correlation is small, and the curves are much more dissimilar. ♦

Several special cases of this model are mentioned by Qian et al. (2008). Each reduces the number of parameters one needs to fit the model. The simplest case assumes $\tau_{t_i,t_j} = \tau$ for $i \neq j$. This is sometimes referred to as the exchangeable model.

Another case (see McMillan et al. (1999)) assumes

$$\tau_{t_i,t_j} = \left(\exp\left\{-(\xi_i^* + \xi_j^*)\right\} I\{i \neq j\}\right)$$

where ξ_i^* and ξ_j^* are positive and $I\{i \neq j\}$ is the indicator function. Note that the McMillan et al. (1999) structure is the same as in the naive use of indicator variables described previously. This was shown to have undesirable properties in general.

A third special case is a Kronecker product structure. Suppose there are J qualitative variables, and the j^{th} qualitative variable has T_j levels. Let $\boldsymbol{t} = (t_1, t_2, \ldots, t_J)^\top$ denote the vector of qualitative variable levels for an input that has qualitative variable j at level t_j for $j = 1, \ldots, J$. A legitimate correlation function is

$$Cor\left[Z_{\boldsymbol{t}_1}(\boldsymbol{x}_1), Z_{\boldsymbol{t}_2}(\boldsymbol{x}_2)\right] = \prod_{j=1}^{J} \tau_{j,t_{1j},t_{2j}} \prod_{i=1}^{d} \exp\left\{-\xi_i \left(x_{1,i} - x_{2,i}\right)^2\right\}$$

where $\boldsymbol{\tau}_j$, the $T_j \times T_j$ matrix with $(r,s)^{th}$ entry $\tau_{j,r,s}$, is positive definite with unit diagonal entries. This corresponds to taking $\boldsymbol{\tau} = \boldsymbol{\tau}_1 \otimes \cdots \otimes \boldsymbol{\tau}_J$, where $\otimes$ is the Kronecker product. This case reduces the number of τ_{t_i,t_j} parameters in the general case of the model given in (2.4.3). It also "imposes" a sort of multiplicative main effects structure on the τ_{t_i,t_j} and in this way takes into account the factorial structure.

Qian et al. (2008) consider additional forms for $\boldsymbol{\tau}$ and for the τ_{t_i,t_j} that assume the levels of the qualitative factor can be organized into similar groups and that allow for ordinal qualitative factors.

The flexibility of the formulation in Qian et al. (2008) makes the model attractive. However, this model makes some assumptions about the different response surfaces determined by t. In particular, in the correlation structure given in (2.4.3), the correlation parameters ξ_k and the process variance σ_Z^2 *are the same for all values of* t. This implies that the "shape" of the local variation as a function of the quantitative variables is the same for all t. If this is not the case, this model may perform worse than simply fitting separate GP models to each curve.

Zhang (2014) describes a method for incorporating qualitative variables into the correlation function that is "equivalent" to the correlation structure in (2.4.3). For

$1 \le p \le T$ define

$$I_p(i) = \begin{cases} 1 \text{ if } i = p \\ 0 \text{ otherwise} \end{cases},$$

and for $1 \le p, q \le T - 1$,

$$W_{p,q}(i) = \begin{cases} I_p(i) + I_q(i) & \text{if } p \ne q \\ I_p(i) & \text{if } p = q \end{cases}.$$

Let

$$Cor\left[Z_{t_1}(\boldsymbol{x}_1), Z_{t_2}(\boldsymbol{x}_2)\right] = \prod_{p,q=1}^{T-1} \exp\left\{-\xi^*_{p,q}\left(W_{p,q}(t_1) - W_{p,q}(t_2)\right)^2\right\} \times \prod_{k=1}^{d} \exp\left\{-\xi_k\,(x_{1,k} - x_{2,k})^2\right\}. \tag{2.4.4}$$

One can show, assuming $\xi^*_{p,q} = \xi^*_{q,p}$ and that the $\tau_{i,j}$ in (2.4.3) are > 0, that for $i \ne j,\ i < T,\ j < T$,

$$-\ell n(\tau_{i,j}) = \xi^*_{i,i} + \xi^*_{j,j} - 4\xi^*_{i,j} + 2\sum_{q=1,q\ne i}^{T-1} \xi^*_{i,q} + 2\sum_{q=1,q\ne j}^{T-1} \xi^*_{j,q}\,;$$

and for $i \ne j,\ i = T,\ j < T$,

$$-\ell n(\tau_{T,j}) = \xi^*_{j,j} + 2\sum_{q=1,q\ne j}^{T-1} \xi^*_{j,q}\,;$$

and for $i \ne j,\ i < T,\ j = T$,

$$-\ell n(\tau_{i,T}) = \xi^*_{i,i} + 2\sum_{q=1,q\ne i}^{T-1} \xi^*_{i,q}\,.$$

Also for $i \ne j,\ i < T,\ j < T$,

$$\xi^*_{i,j} = \frac{1}{4}\left(\ell n(\tau_{i,j}) - \ell n(\tau_{T,j}) - \ell n(\tau_{i,T})\right)\,;$$

and for $i < T$,

$$\xi^*_{i,i} = -\frac{1}{2}\sum_{q=1,q\ne i}^{T} \ell n(\tau_{i,q}) + \frac{1}{2}\sum_{q=1,q\ne i}^{T-1} \ell n(\tau_{T,q})\,.$$

Thus when the $\tau_{i,j}$ are > 0 in (2.4.3), there is a one-to-one correspondence between the $\tau_{i,j}, i \ne j$, and the $\xi^*_{p,q}\,,\ p < T,\ q < T$, in the sense that given the $\tau_{i,j}$ it is possible to determine the corresponding $\xi^*_{p,q}$ and vice versa.

The correlation function in (2.4.4) is the separable version of the Gaussian correlation function (2.2.8) expressed as a function of the $W_{p,q}(\cdot)$ (representing the qualitative variables) and $\boldsymbol{x}$ (representing the quantitative variables). Chapter 3 describes software for fitting GP models with quantitative inputs in which the Gaussian correlation function is assumed, and one advantage of (2.4.4) is that this same software can be used to fit GP models with both qualitative and quantitative inputs.

Another way to incorporate qualitative variables into (2.4.2) is inspired by how one can characterize the multivariate normal distribution (and more generally by models for multiple correlated GPs). Let $N_1(\boldsymbol{x}), N_2(\boldsymbol{x}), \ldots, N_S(\boldsymbol{x})$ be S independent, identically distributed mean zero, stationary GPs with variance σ_Z^2. Assume each satisfies

$$Cor\,[N_i(\boldsymbol{x}_1), N_i(\boldsymbol{x}_2)] = \prod_{k=1}^{d} \exp\left\{-\xi_k\,(x_{1,k} - x_{2,k})^2\right\}$$

for $1 \le i \le S$. Assume for $1 \le t \le T$

$$Z_t(\boldsymbol{x}) = \sum_{i=1}^{S} a_{t,i}\, N_i(\boldsymbol{x})$$

for some constants $a_{t,i}$. Then

$$(Z_1(\boldsymbol{x}), \ldots, Z_T(\boldsymbol{x}))^\top = \boldsymbol{A}(N_1(\boldsymbol{x}), \ldots, N_S(\boldsymbol{x}))^\top$$

where $\boldsymbol{A}$ is the $T \times S$ matrix with $a_{i,j}$ as its $(i, j)^{th}$ entry.

This yields the Qian et al. (2008) model provided $\boldsymbol{\tau} = \boldsymbol{A}\boldsymbol{A}^\top$. Qian et al. (2008) use this representation to prove that $\boldsymbol{\tau}$ must be a positive definite symmetric matrix with unit diagonal entries for the correlation structure in their model to be valid. But, by analogy with multivariate normal methods, there is much more that can be done with this representation.

The exchangeable model is produced by assuming

$$Z_i(\boldsymbol{x}) = \sqrt{\tau}\, N_1(\boldsymbol{x}) + \sqrt{1-\tau}\, N_{i+1}(\boldsymbol{x})\,.$$

This indicates that the $Y(\boldsymbol{x}, i)$ in (2.4.2) are composed of a common overall trend (the $N_1(\boldsymbol{x})$ term) and independent realizations of a "treatment effect" trend (the $N_{i+1}(\boldsymbol{x})$ terms). Both the overall trend and treatment effect trends are of the same magnitude for each $Y(\boldsymbol{x}, i)$. Notice that the exchangeable model could be interpreted as having a one-way ANOVA structure.

To represent the Kronecker product structure, let

$$\boldsymbol{N}^j(\boldsymbol{x}) = (N_1^j(\boldsymbol{x}), \ldots, N_{T_j}^j(\boldsymbol{x}))^\top$$

where the $N_i^j(\boldsymbol{x})$ components are independent, identically distributed mean zero, stationary GPs with process variance σ_Z^2. Let $\boldsymbol{A}_j$ be a $T_j \times T_j$ matrix satsfying $\boldsymbol{\tau}_j = \boldsymbol{A}_j\boldsymbol{A}_j^\top$. Then in our general formulation $\boldsymbol{A}(N_1(\boldsymbol{x}), \ldots, N_S(\boldsymbol{x}))^\top$ becomes

$$\left(\bigotimes_{j=1}^{J} \boldsymbol{A}_j \right) \left(\bigotimes_{j=1}^{J} \boldsymbol{N}^j(\boldsymbol{x}) \right).$$

One can impose a factorial structure on the $Y(\boldsymbol{x}, i)$ in (2.4.2). For example, suppose the $Y(\boldsymbol{x}, i)$ are determined by two factors F and G with f and g levels, respectively. Suppose $Y(\boldsymbol{x}, i)$ corresponds to F at level ϕ and G at level γ. Let

$$Z_i(\boldsymbol{x}) = a_i^{\mu} N^{\mu}(\boldsymbol{x}) + a_i^{F} N_{\phi}^{F}(\boldsymbol{x}) + a_i^{G} N_{\gamma}^{G}(\boldsymbol{x}),$$

where $N^{\mu}(\boldsymbol{x})$ is an overall mean effect (trend), $N_{\phi}^{F}(\boldsymbol{x})$ is the effect of level ϕ of F, and $N_{\gamma}^{G}(\boldsymbol{x})$ is the effect of level γ of G. This looks like a two-factor main effects model.

If one does not require $\boldsymbol{A}$ to satisfy $\boldsymbol{\tau} = \boldsymbol{A}\boldsymbol{A}^{\top}$, then the formulation

$$(Z_1(\boldsymbol{x}), \ldots, Z_T(\boldsymbol{x}))^{\top} = \boldsymbol{A}(N_1(\boldsymbol{x}), \ldots, N_s(\boldsymbol{x}))^{\top}$$

allows the $Z_i(\boldsymbol{x})$ to have different variances.

Another application of this representation of the Qian et al. (2008) model is to an analysis that is suggestive of factor analysis. Estimate the matrix $\boldsymbol{\tau}$ and find a parsimonious matrix $\boldsymbol{A}$ so that $\boldsymbol{\tau} = \boldsymbol{A}\boldsymbol{A}^{\top}$. Note that $\boldsymbol{A}$ is only determined up to multiplication by an orthogonal matrix. The form of $\boldsymbol{A}$ may suggest some sort of factorial structure, or perhaps that the $Y(\boldsymbol{x}, i)$ depend mostly on a relatively small number of the $N_j(\boldsymbol{x})$.

Remark: In the hypersphere parameterization of $\boldsymbol{\tau}$ discussed in Zhou et al. (2011), they essentially use $\boldsymbol{\tau} = \boldsymbol{A}\boldsymbol{A}^{\top}$ with $\boldsymbol{A}$ being lower triangular, guaranteeing the uniqueness of $\boldsymbol{A}$.

Example 2.10. Zhou et al. (2011) consider a simple example involving three curves on $x \in [0, 1]$ having a single qualitative variable t with three levels:

$$y(x,t) = \begin{cases} \cos\left(\dfrac{6.8\pi x}{2}\right) & \text{if } t = 1 \\ -\cos\left(\dfrac{7\pi x}{2}\right) & \text{if } t = 2 \\ \cos\left(\dfrac{7.2\pi x}{2}\right) & \text{if } t = 3 \end{cases}.$$

These curves are displayed in Fig. 2.12.

Both the functional forms of the three curves and Fig. 2.12 suggest that the three are highly pairwise correlated and that the GP model,

$$Y(\boldsymbol{x}, t) = \beta_0 + Z_t(\boldsymbol{x})$$

with

$$Z_t(\boldsymbol{x}) = a_t N_1(\boldsymbol{x}), \quad t = 1, 2, 3,$$

and $a_1 = a_3 = -a_2$, may adequately describe these curves. ♦

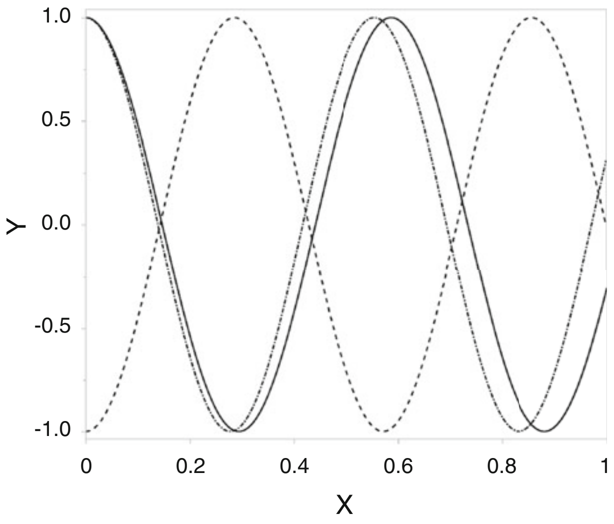


Fig. 2.12 Function curves at $t = 1$ (solid line), $t = 2$ (dashed line), and $t = 3$ (dashed dotted line)

This section concludes by describing a hierarchical model approach for incorporating qualitative variables. Again let $y(\boldsymbol{x}, t)$ denote a deterministic output that depends on the quantitative input $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)^\top$ and a single qualitative input $t \in \{1, \ldots, T\}$. In brief, Han et al. (2009a) assume that the $y(\boldsymbol{x}, t)$ have "similar" dependences on $\boldsymbol{x}$ for different levels t.

In more detail, Han et al. (2009a) apply a model that they term the hierarchical quantitative-qualitative variable (HQQV) model to describe $y(\boldsymbol{x}, t)$. The first stage of the HQQV model assumes that (2.4.2) holds given β_t, σ^2_t, and correlation parameters $\rho_{t,k}$, $t \in \{1, \ldots, T\}$, and $k \in \{1, \ldots, d\}$ where the $Z_t(\cdot)$ are mutually independent with

$$Cor\,[Z_t(\boldsymbol{x}_1), Z_t(\boldsymbol{x}_2)] = \prod_{k=1}^{d} \rho_{t,k}^{(x_{1,k}-x_{2,k})^2}.$$

Recall ρ_{tj} determines the smoothness of $y(\boldsymbol{x}, t)$ for the j^{th} input x_j. They choose $\beta_1, \ldots, \beta_T$ to have a non-informative prior distribution proportional to 1. Han et al. (2009a) scale the responses in each (qualitative input) level to have standard error 1. They choose $\sigma^2_1, \ldots, \sigma^2_T$ to be independently and identically distributed as inverse gamma with parameters selected to have support in an interval about 1. In the spirit of Christiansen and Morris (1997) and Wallstrom (2007), they assume, for each input $k \in \{1, \ldots, d\}$, that $\rho_{1,k}, \ldots, \rho_{T,k}$ are independently and identically distributed as beta(α_k, γ_k). The intuition behind this is that the dependence structure of the k^{th} input is "similar" for all surfaces $y(\boldsymbol{x}, t)$. They use a two-step procedure to obtain empirical estimates of the (α_k, γ_k).

First, they estimate $\rho_{1,k}, \ldots, \rho_{T,k}$ separately based on the data at each level t, $t = 1, \ldots, T$, yielding $\widehat{\rho}_{1,k}, \ldots, \widehat{\rho}_{T,k}$. Second, they chose parameters α_k and γ_k so that the mean of the Beta(α_k, γ_k) prior is

$$M = \max\{\widehat{\rho}_{1,k}, \ldots, \widehat{\rho}_{T,k}\}$$

but truncated to have a lower bound of 0.005 and an upper bound of 0.995, i.e.,

$$\frac{\alpha_k}{\alpha_k + \gamma_k} = median\,\{0.005, M, 0.995\}.$$

In a similar way, they set the variance of beta(α_k, γ_k) equal to the sample variance of $\widehat{\rho}_{1,k}, \ldots, \widehat{\rho}_{T,k}$ but bounded above by 0.004. These two equations determine α_k and γ_k.

The reasoning behind this empirical prior for $\{\rho_{1,k}, \ldots, \rho_{T,k}\}$ is as follows. When the experimental design is not space-filling, or when the number of training data points in level t differs from those in the other levels, an estimate of $\rho_{t,k}$ can be close to 0. If $\widehat{\rho}_{t,k}$ is near 0, then the prediction of the output at level t will converge quickly, in dimension k for the quantitative input, to the process mean, and thus the prediction errors can be undesirably large. With the assumption that the correlation structures of the processes at all levels of the qualitative input are similar, this conservatively avoids this problem by letting $M = \max\{\widehat{\rho}_{1,k}, \ldots, \widehat{\rho}_{T,k}\}$ be the mean of Beta(α_k, γ_k) when $M \in [0.005, 0.995]$. Han et al. (2009a) also assume that the $(\beta_1, \ldots, \beta_T)$, $(\sigma_1^2, \sigma_2^2, \ldots, \sigma_T^2)$, and $(\rho_{1,k}, \ldots, \rho_{T,k})$ are mutually independent. They describe how to perform prediction at unknown inputs based on this model.

2.5 Models for Multivariate and Functional Simulator Output

2.5.1 Introduction

This subsection describes GP models $\boldsymbol{Y}(\boldsymbol{x}) = (Y_1(\boldsymbol{x}), Y_2(\boldsymbol{x}), \ldots, Y_m(\boldsymbol{x}))$ for multivariate output $\boldsymbol{y}(\boldsymbol{x}) = (y_1(\boldsymbol{x}), y_2(\boldsymbol{x}), \ldots, y_m(\boldsymbol{x}))$. Section 2.4 described one setting where such a model is necessary, namely, that in which the data have mixed quantitative and qualitative inputs and the levels of the qualitative inputs correspond to the m outputs. Another setting requiring a multivariate model was described in Example 1.6 of Sect. 1.2 and is reviewed here.

Example 1.6 (Continued). As an example of multivariate data, recall the 3-D finite element simulator of Ong et al. (2008) that produced $m = 4$ related outputs $y_1(\boldsymbol{x})$, $y_2(\boldsymbol{x})$, $y_3(\boldsymbol{x})$, and $y_4(\boldsymbol{x})$ each of which measured an aspect of acetabular cup stability under loading (`total potential ingrowth area`; `change in gap volume`; `gap volume`; and `cup relative motion`). There were 12 inputs to this model; they can be grouped into those inputs that determine the mechanical design of the cup, those that describe the surgical skill with which the cup is inserted, and inputs that describe the environment where it will operate.

Figure 2.13 displays all six pairs of the $m = 4$ outputs corresponding to 39 runs of the code. Several of the outputs are highly related, for example, the plot of `change in gap volume` $\times$ `cup relative motion` shows a strong linear trend. ♦

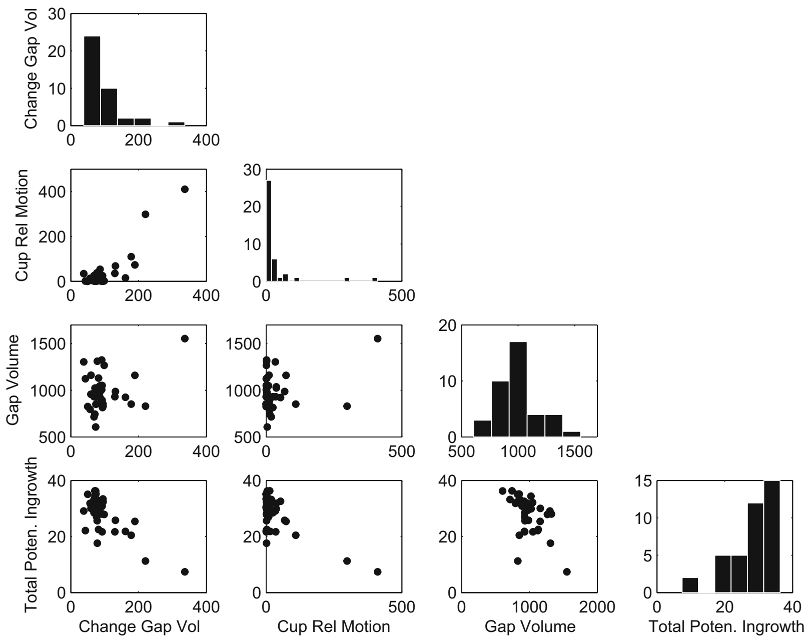


Fig. 2.13 Matrix scatterplot of $m = 4$ outputs for Example 1.6 based on $n = 39$ runs

"Functional data" are a third type of data that are essentially multivariate; data are functional provided each output is a function, say of the argument $\boldsymbol{t} \in \mathcal{T}$. The notation used to describe functional output is $y(\boldsymbol{x}, \boldsymbol{t})$, where $\boldsymbol{x}$ denotes controllable inputs, while the computed response corresponding to a given $\boldsymbol{x}_0$, $y(\boldsymbol{x}_0, \boldsymbol{t})$, is a function of $\boldsymbol{t} \in \mathcal{T}$. For example, Fig. 1.14 shows the concentration of a chemical pollutant over a location $\times$ time grid (the $\mathcal{T}$), for a specific selection of the inputs (M, D, L, T) that describe the characteristics of the chemical spill. Functional data are ordinarily computed at a finite $\boldsymbol{t}$ grid.

A common way to reduce functional data to multivariate output is to model the functional output only at a few physically important "landmark" values of $\boldsymbol{t} \in \mathcal{T}$. Another method to reduce the highly dependent (in $\boldsymbol{t}$) functional outputs to low-dimensional multivariate data is by using an appropriate basis; the latter method will be illustrated in Sect. 2.5.4.

In practice, two types of multivariate training data are collected for prediction and other inferences. The first type of training data is when *all* m outputs have been evaluated at n inputs $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$; the second type is when only a *subset* of the m

outputs have been evaluated at each of the n inputs. The second case occurs when there are multiple codes not all of which need be run at the same set of inputs.

The outputs for the first case will be denoted by $\boldsymbol{y}(\boldsymbol{x}_1), \ldots, \boldsymbol{y}(\boldsymbol{x}_n)$ and model values by $\boldsymbol{Y}^1 \equiv \boldsymbol{Y}(\boldsymbol{x}_1), \ldots, \boldsymbol{Y}^n \equiv \boldsymbol{Y}(\boldsymbol{x}_n)$. The notation for second case is more complex. As in the first case, let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ denote the set of *all inputs at which any code runs have been made*. Let $1 \leq \ell_1^i < \cdots < \ell_{m_i}^i \leq m$ denote the indices of the output functions at which $\boldsymbol{x}_i$ has been calculated, $i = 1, \ldots, n$. For example, it might be that only $y_1(\boldsymbol{x}_i)$, $y_3(\boldsymbol{x}_i)$, and $y_5(\boldsymbol{x}_i)$ have been computed for $\boldsymbol{x}_i$ when there are $m = 6$ possible outputs; thus $m_i = 3$ and $\ell_1^i = 1$, $\ell_2^i = 3$, and $\ell_3^i = 5$. Denote the vector of calculated outputs at $\boldsymbol{x}_i$ by $\boldsymbol{y}^i = (y_{\ell_1^i}(\boldsymbol{x}_i), \ldots, y_{\ell_{m_i}^i}(\boldsymbol{x}_i))$, for $i = 1, \ldots, n$. The model values associated with the $\boldsymbol{y}^i$ runs will be denoted by $\boldsymbol{Y}^i$, $i = 1, \ldots, n$.

The multivariate GP models described below for $\boldsymbol{Y}(\boldsymbol{x}) = (\boldsymbol{Y}^1, \ldots, \boldsymbol{Y}^n)$ yield valid multivariate normal distributions; this means that $\boldsymbol{Y}(\boldsymbol{x})$ has a positive semi-definite covariance matrix. (An arbitrarily selected parametric form for the individual cross-covariances

$$Cov\left[Y_{\ell^i}(\boldsymbol{x}_i), Y_{\ell^q}(\boldsymbol{x}_q)\right], \text{ for } i \neq q,$$

need not produce a positive semi-definite covariance matrix for $\boldsymbol{Y}(\boldsymbol{x})$.) In addition to the covariance matrix being positive semi-definite, it is desired that a valid GP for $\boldsymbol{Y}(\boldsymbol{x})$ embody whatever smoothness properties are known marginally about each response $y_i(\cdot)$, $1 \leq i \leq m$, and also what is known about the *dependencies* among the m outputs. The former might mean that the model might have a certain degree of smoothness and contain terms that describe specified large-scale trends in certain inputs. The latter, for example, might require that competing outputs $y_i(\boldsymbol{x})$ and $y_j(\boldsymbol{x})$ be negatively correlated because larger values of $y_i(\boldsymbol{x})$ are associated with lower values of $y_j(\boldsymbol{x})$.

Sections 2.5.2 and 2.5.3 describe constructive methods that produce valid multivariate GP models $(Y_1(\boldsymbol{x}), Y_2(\boldsymbol{x}), \ldots, Y_m(\boldsymbol{x}))$. Section 2.5.4 introduces basis methods for functional simulator data.

2.5.2 Modeling Multiple Outputs

This section describes GP models for $\boldsymbol{Y}(\boldsymbol{x})$ that have marginal distributions of the form

$$Y_i(\boldsymbol{x}) = \boldsymbol{f}_i^\top(\boldsymbol{x})\boldsymbol{\beta}_i + Z_i(\boldsymbol{x}), \tag{2.5.1}$$

for $i = 1, \ldots, m$ where $Z_i(\cdot)$ is a mean zero GP whose other properties depend on the details of the definition of $\boldsymbol{Z}(\boldsymbol{x}) = (Z_1(\boldsymbol{x}), \ldots, Z_m(\boldsymbol{x}))$. The linear model $\boldsymbol{f}_i^\top(\boldsymbol{x})\boldsymbol{\beta}_i$ represents the global trend of the $Y_i(\boldsymbol{x})$ process; here $\boldsymbol{f}_i(\boldsymbol{x}) = (f_{i,1}(\boldsymbol{x}), f_{i,2}(\boldsymbol{x}), \ldots, f_{i,p_i}(\boldsymbol{x}))^\top$ is a $p_i \times 1$ vector of *known* regression functions and $\boldsymbol{\beta}_i = (\beta_{i,1}, \ldots, \beta_{i,p_i})^\top$ is a $p_i \times 1$ vector of *unknown* regression parameters. Letting $\boldsymbol{\beta} = (\boldsymbol{\beta}_1^\top, \ldots, \boldsymbol{\beta}_m^\top)^\top$ denote the $(\sum_1^m p_i) \times 1$ vector of all regression parameters, the mean of $\boldsymbol{Y}(\boldsymbol{x})$ can be written in matrix form as

$$E[\boldsymbol{Y}(\boldsymbol{x})] = \boldsymbol{F}(\boldsymbol{x})\boldsymbol{\beta} \equiv \begin{bmatrix} \boldsymbol{f}_1^\top(\boldsymbol{x}) & \boldsymbol{0}_{1\times p_2} & \cdots & \boldsymbol{0}_{1\times p_m} \\ \boldsymbol{0}_{1\times p_1} & \boldsymbol{f}_2^\top(\boldsymbol{x}) & \cdots & \boldsymbol{0}_{1\times p_m} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0}_{1\times p_1} & \boldsymbol{0}_{1\times p_2} & \cdots & \boldsymbol{f}_m^\top(\boldsymbol{x}) \end{bmatrix} \boldsymbol{\beta}\,. \tag{2.5.2}$$

An important special case of (2.5.2) is when all the outputs use the *same* regression functions but with component-specific regression parameters, i.e., $\boldsymbol{f}_1(\boldsymbol{x}) = \cdots = \boldsymbol{f}_m(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x})$ and thus $p_1 = \cdots = p_m = p$, say. An important special case of the common regression mean model is $\boldsymbol{f}_1(\boldsymbol{x}) = \cdots = \boldsymbol{f}_m(\boldsymbol{x}) = 1$ which produces $Y_i(\boldsymbol{x})$ with constant but component-specific means.

This subsection focuses on the *nonseparable linear model of coregionalization* (NLMC) for $\boldsymbol{Z}(\boldsymbol{x})$. The NLMC model is extremely flexible; it is used in both computer experiments (Svenson and Santner (2016) and Fricker et al. (2013)) and geostatistical applications of spatial statistics (Banerjee et al. (2008)).

An NLMC model for $\boldsymbol{Z}(\boldsymbol{x})$ has the following representation. Select a *symmetric* $m \times m$ *positive-definite* matrix $\boldsymbol{A}$ then set

$$\boldsymbol{Z}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{N}(\boldsymbol{x}) \tag{2.5.3}$$

where, as in Sect. 2.4, $N_1(\boldsymbol{x}), N_2(\boldsymbol{x}), \ldots, N_m(\boldsymbol{x})$ are m independent, mean zero, unit variance stationary GPs, but $N_i(\boldsymbol{x})$ is allowed to have its own correlation function $R_i(\boldsymbol{h})$, $1 \le i \le m$.

Combining the mean and covariance assumptions from the previous paragraphs, the cross-covariance matrix of

$$\boldsymbol{Y}(\boldsymbol{x}) = \boldsymbol{F}(\boldsymbol{x})\boldsymbol{\beta} + \boldsymbol{A}\boldsymbol{N}(\boldsymbol{x}) \tag{2.5.4}$$

corresponding to $\boldsymbol{x}_1 \neq \boldsymbol{x}_2$ is the $m \times m$ matrix

$$Cov\,[\boldsymbol{Y}(\boldsymbol{x}_1), \boldsymbol{Y}(\boldsymbol{x}_2)] = \boldsymbol{A}\boldsymbol{D}\boldsymbol{A}^\top \tag{2.5.5}$$

where $\boldsymbol{D} = diag(R_1(\boldsymbol{x}_1 - \boldsymbol{x}_2), \ldots, R_m(\boldsymbol{x}_1 - \boldsymbol{x}_2))$ denotes the diagonal matrix

$$\begin{bmatrix} R_1(\boldsymbol{x}_1 - \boldsymbol{x}_2) & 0 & \cdots & 0 \\ 0 & R_2(\boldsymbol{x}_1 - \boldsymbol{x}_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & R_m(\boldsymbol{x}_1 - \boldsymbol{x}_2) \end{bmatrix}.$$

In particular, the covariance matrix of $\boldsymbol{Y}(\boldsymbol{x})$ is

$$Cov\,[\boldsymbol{Y}(\boldsymbol{x}), \boldsymbol{Y}(\boldsymbol{x})] = \boldsymbol{A}\boldsymbol{A}^\top\,. \tag{2.5.6}$$

The literature contains a number of important special cases of (2.5.3) corresponding to particular forms of $\boldsymbol{A}$ and assumptions about $\{R_i(\boldsymbol{h})\}_{i=1}^m$. For example, when

$\boldsymbol{A} = diag(a_{11}, \ldots, a_{mm})$ is diagonal then the cross-covariance matrix (2.5.5) reduces to

$$Cov\left[\boldsymbol{Y}(\boldsymbol{x}_1), \boldsymbol{Y}(\boldsymbol{x}_2)\right] = diag\left(a_{11}^2 R_1(\boldsymbol{x}_1 - \boldsymbol{x}_2), \ldots, a_{mm}^2 R_m(\boldsymbol{x}_1 - \boldsymbol{x}_2)\right),$$

showing that $\boldsymbol{Y}(\boldsymbol{x})$ has *independent* components and $Var[Y_i(\boldsymbol{x})] = a_{ii}^2$, $1 \le i \le m$. Another special case of the NLMC model is when $R_1(\boldsymbol{h}) = \cdots = R_m(\boldsymbol{h}) = R(\boldsymbol{h})$, in which case the cross-covariance matrix is

$$\begin{aligned} Cov\left[\boldsymbol{Y}(\boldsymbol{x}_1), \boldsymbol{Y}(\boldsymbol{x}_2)\right] &= \boldsymbol{A}\, diag(R(\boldsymbol{x}_1 - \boldsymbol{x}_2), \ldots, R(\boldsymbol{x}_1 - \boldsymbol{x}_2))\, \boldsymbol{A}^\top \\ &= R(\boldsymbol{x}_1 - \boldsymbol{x}_2)\, \boldsymbol{A}\boldsymbol{A}^\top \end{aligned}$$

which is the separable covariance model of Conti and O'Hagan (2010).

Arguably the most critical limitation of the NLMC model is seen in the covariance formula (2.5.6); the association among $Y_1(\boldsymbol{x}), \ldots, Y_m(\boldsymbol{x})$ is the *same* for *all* points $\boldsymbol{x}$ of the input space. More flexible models that allow $\boldsymbol{A}$ in (2.5.4) to depend on $\boldsymbol{x}$, and hence the relationship among $Y_1(\boldsymbol{x}), \ldots, Y_m(\boldsymbol{x})$ to vary with $\boldsymbol{x}$, have been considered in the literature (Gelfand et al. (2004), Chen et al. (2014a)).

Example 2.11. Kennedy and O'Hagan (2000) used a spatial autocorrelation model to describe the output of a set of multi-fidelity computer codes $y_1(\boldsymbol{x}), \ldots, y_m(\boldsymbol{x})$. Let $i = m$ denote the highest fidelity code and *smaller* values of i denote successively *lower* fidelity (less complex) codes. Let $Y_i(\boldsymbol{x})$ denote the process model for the i^{th} level of the code, $i = 1, \ldots, m$. The object is to predict $y_m(\boldsymbol{x})$ using the runs from all fidelities.

The Kennedy and O'Hagan (2000) model views the observed data as a draw from $\boldsymbol{Y}(\boldsymbol{x}) = (Y_1(\boldsymbol{x}), \ldots, Y_m(\boldsymbol{x}))$ which relates the output at level i, $y_i(\boldsymbol{x})$, to the immediate lower fidelity output $y_{i-1}(\boldsymbol{x})$ plus a refinement, i.e.,

$$Y_i(\boldsymbol{x}) = \rho_{i-1} Y_{i-1}(\boldsymbol{x}) + \varDelta_i(\boldsymbol{x}), \quad i = 2, \ldots, m, \tag{2.5.7}$$

where ρ_{i-1}, $0 < \rho_{i-1} < 1$, is a constant that describes the amount of common behavior in the lower and higher fidelity codes and $\varDelta_i(\boldsymbol{x})$ is a process, independent of $Y_{i-1}(\boldsymbol{x})$, that describes the enhancements of $y_i(\boldsymbol{x})$ over $y_{i-1}(\boldsymbol{x})$. This model can be embellished by allowing a separate regression for each stage of the model.

Assuming that $y_1(\boldsymbol{x})$ can be described as a draw from $\varDelta_1(\boldsymbol{x})$, the autoregressive model (2.5.7) is of the form (2.5.4). A small amount of algebra shows

$$Y_i(\boldsymbol{x}) = \prod_{j=1}^{i-1} \rho_j \varDelta_1(\boldsymbol{x}) + \sum_{\ell \le i-1} \left\{ \prod_{j=\ell}^{i-1} \rho_j \varDelta_\ell(\boldsymbol{x}) \right\} + \varDelta_i(\boldsymbol{x}) \tag{2.5.8}$$

for $i = 2, \ldots, m$. Equation (2.5.8) implicitly defines the $\boldsymbol{A}$ matrix in (2.5.4) in terms of $(\varDelta_1(\boldsymbol{x}), \ldots, \varDelta_m(\boldsymbol{x}))$. ■

2.5.3 Other Constructive Models

Constructive approaches have also been used to form process models for prediction in other settings. Two of these applications are environmental science (Ver Hoef and Barry (1998); Higdon (1998)) and computer experiments (Morris et al. (1993); Mitchell et al. (1994)).

Models for environmental science applications typically describe one or more unknown smooth surfaces which are observed with measurement error. As an example, Ver Hoef and Barry (1998) modeled the *observed* spatial processes to be moving averages over white noise processes. Let $y(\boldsymbol{x}) = (y_1(\boldsymbol{x}), \ldots, y_m(\boldsymbol{x}))$ denote the observed data at input $\boldsymbol{x}$. To describe their $\boldsymbol{Y}(\boldsymbol{x})$ model for $y(\boldsymbol{x})$, let $\{W_0(\boldsymbol{x}), W_1(\boldsymbol{x}), \ldots, W_m(\boldsymbol{x})\}$ denote mutually independent, mean zero, white noise processes. For $i = 1, \ldots, m$ set

$$Z_i(\boldsymbol{w}) = \sqrt{1-\rho_i^2}\, W_i(\boldsymbol{w}) + \rho_i W_0(\boldsymbol{w} - \boldsymbol{\varsigma}_i),$$

where $-1 \le \rho_i \le 1$ and $\boldsymbol{\varsigma}_i$ is a shift parameter that determines the cross-correlation function among the $\{Z_i\}_{i=1}^m$ processes as seen in the following formula. Fixing $i_1 \neq i_2$ calculation gives

$$Cor\left[Z_{i_1}(\boldsymbol{w} + \boldsymbol{\varsigma}_{i_1}), Z_{i_2}(\boldsymbol{w} + \boldsymbol{\varsigma}_{i_2})\right] = \rho_{i_1}\rho_{i_2}\,,$$

but is zero otherwise; thus nonzero cross-correlations can only occur when inputs are separated by the amount $\boldsymbol{\varsigma}_{i_1} - \boldsymbol{\varsigma}_{i_2}$. Ver Hoef and Barry (1998) assume the observed $y_i(\boldsymbol{x})$ is a draw from

$$Y_i(\boldsymbol{x}) = \mu_i + \int f_i(\boldsymbol{w} - \boldsymbol{x})\, Z_i(\boldsymbol{w})\, d\boldsymbol{w} + \sigma_i\, \xi_i(\boldsymbol{x})\,,$$

which is a process mean μ_i *plus* the integrated $Z_i(\boldsymbol{x})$ processes with respect to a (moving average) function $f_i(\boldsymbol{h})$ *plus* a measurement error, $i = 1, \ldots, m$. The measurement error, $\sigma_i\, \xi_i(\boldsymbol{x})$, is based on the zero mean, unit variance white noise processes $\xi_i(\boldsymbol{x})$ which are taken to be mutually independent, for $1 \le i \le m$, and also independent of $\boldsymbol{Z}(\boldsymbol{x})$; thus σ_i^2 is the measurement error variance. Each $f_i(\boldsymbol{h})$ is taken to be square integrable which ensures that $Y_i(\boldsymbol{x})$, $i = 1, \ldots, m$, is second-order stationary. Ver Hoef and Barry (1998) found that it is possible to reproduce many commonly used variogram models with this type of moving average construction.

Turning attention to computer experiments applications, consider a setting in which the simulator code produces $y(\boldsymbol{x})$ *and* the first partial derivatives ("adjoints") of $y(\boldsymbol{x})$. Suppose that the input space for the code is denoted $\mathcal{X} \subset \mathbb{R}^d$ and that $y(\boldsymbol{x})$ has first partial derivatives of all orders. Let

$$y^{(j)}(\boldsymbol{x}) = \partial y(\boldsymbol{x}) / \partial x_j$$

denote the jth partial derivative of $y(\boldsymbol{x})$ for $1 \le j \le d$. In the general notation introduced above, there are $m = 1 + d$ outputs and $y_1(\boldsymbol{x}) = y(\boldsymbol{x}), y_2(\boldsymbol{x}) = y^{(1)}(\boldsymbol{x}), \ldots$, and $y_m(\boldsymbol{x}) = y^{(d)}(\boldsymbol{x})$.

Morris et al. (1993) and Mitchell et al. (1994) consider a multivariate model for the setting of the previous paragraph in which the prior information about $y(\boldsymbol{x})$ is specified by a Gaussian process $Y(\cdot)$ and the prior about the partial derivatives $y^{(j)}(\boldsymbol{x})$ is obtained by considering the "derivative" processes of $Y(\cdot)$. Suppose that the prior for $y(\boldsymbol{x})$ is

$$Y(\boldsymbol{x}) = \boldsymbol{f}^\top(\boldsymbol{x})\boldsymbol{\beta} + Z(\boldsymbol{x}),$$

where each component of $\boldsymbol{f}^\top(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_p(\boldsymbol{x}))$ has first partial derivatives with respect to all components x_j, $1 \le j \le d$, and $Z(\boldsymbol{x})$ is a stationary Gaussian process with zero mean, variance σ_Z^2, and separable Gaussian correlation function

$$R(\boldsymbol{w}) = \prod_{j=1}^{d} \exp\left\{-\xi_j w_j^2\right\}.$$

A "natural" model for the partial derivative $y^{(j)}(\boldsymbol{x})$ is

$$Y^{(j)}(\boldsymbol{x}) = \lim_{h \to 0} \frac{Y(x_1, \ldots, x_{j-1}, x_j + h, x_{j+1}, \ldots, x_d) - Y(\boldsymbol{x})}{h}, \tag{2.5.9}$$

which exists under differentiability conditions for $R(\cdot)$ that are satisfied for the product of Gaussian correlation functions (Parzen (1962)). The fact that linear combinations of multivariate normal random variables are again multivariate normal suggests that the limit (2.5.9) will have the same behavior, which applies here. Under appropriate conditions, the "partial derivative" process $Y^{(j)}(\boldsymbol{x})$ has mean equal to $\frac{\partial \boldsymbol{f}^\top(\boldsymbol{x})\boldsymbol{\beta}}{\partial x_j} = \sum_{\ell=1}^{p} \beta_\ell \, \partial f_\ell(\boldsymbol{x})/\partial x_j$. To complete the statement of the joint distribution of $(Y(\boldsymbol{x}), Y^{(1)}(\boldsymbol{x}), \ldots, Y^{(d)}(\boldsymbol{x}))$, formulas are required for the cross-covariances between $Y(\boldsymbol{x})$ and each $Y^{(j)}(\boldsymbol{x})$. General formulas for the cross-covariances are known even when the $Y(\boldsymbol{x})$ process need not be stationary but has covariance function $Cov[Y(\boldsymbol{x}_1), Y(\boldsymbol{x}_2)] = \sigma_Z^2 R(\boldsymbol{x}_1, \boldsymbol{x}_2)$; these are

$$Cov\left[Y(\boldsymbol{x}_1), Y^{(j)}(\boldsymbol{x}_2)\right] = \sigma_Z^2 \frac{\partial R(\boldsymbol{x}_1, \boldsymbol{x}_2)}{\partial x_{2,j}} \tag{2.5.10}$$

for $1 \le j \le d$ and

$$Cov\left[Y^{(i)}(\boldsymbol{x}_1), Y^{(j)}(\boldsymbol{x}_2)\right] = \sigma_Z^2 \frac{\partial^2 R(\boldsymbol{x}_1, \boldsymbol{x}_2)}{\partial x_{1,i} \, \partial x_{2,j}} \tag{2.5.11}$$

for $1 \le i \le j \le d$ (Morris et al. (1993)).

2.5.4 Models for Simulators Having Functional Output

Examples 1.7 and 1.8 illustrate functional output. Example 1.1, on the temporal evolution of a fire in a closed room, is another example whose output is fundamentally functional. This is because the ASETB simulator computes both temperature and plume height as a *function of the time from the initiation of the fire*. Figure 2.14 shows the rise in temperature for one of the fire scenarios as a function of time. In the initial description of the Example 1.1 output, a landmark value was selected from the *plume height* × *time* curve (the time at which the smoke plume descended to 5 ft above the fire source).

In the general case, let $\mathcal{X} \subset \mathbb{R}^d$ denote the space of controllable inputs $\boldsymbol{x}$ and $\mathcal{T} \subset \mathbb{R}^m$ the domain of the functional output which is assumed to be a finite set. For a fixed $\boldsymbol{x}$, let $y(\boldsymbol{x}) = \{y(\boldsymbol{x}; \boldsymbol{t}) \ : \ \boldsymbol{t} \in \mathcal{T} \subset \mathbb{R}^m\}$ denote the m code outputs. Multivariate data is typically lower dimensional with "small" m than is functional data where m can be very "large."

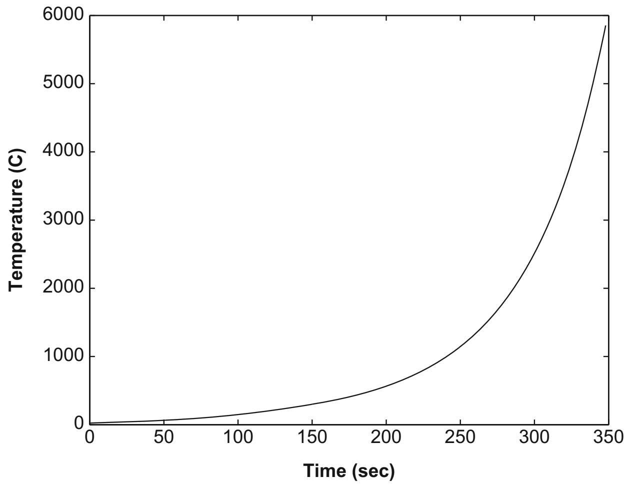


Fig. 2.14 ASETB simulation of the *Temperature* × *Time* curve of a contained fire in a room with characteristics: heat loss fraction = 0.75375, fire height = 1.87 ft, room height = 8.15 ft, and room area = 205.69 ft^2

The primary idea used in modeling functional output $y(\boldsymbol{x})$ is already seen in Fig. 2.14—$y(\boldsymbol{x}; \boldsymbol{t}_1), \ldots, y(\boldsymbol{x}; \boldsymbol{t}_m)$ are closely related. Typical multivariate data such as those plotted in Fig. 2.13 need not be inherently ordered and are typically less closely related than functional data.

Thus most papers in the literature analyze functional data by selecting a set of p_b basis vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{p_b}$ where each $\boldsymbol{b}_i$ is $m \times 1$ and approximate

$$\boldsymbol{y}(\boldsymbol{x}) = \begin{pmatrix} y(\boldsymbol{x}; \boldsymbol{t}_1) \\ \vdots \\ y(\boldsymbol{x}; \boldsymbol{t}_m) \end{pmatrix} \approx \sum_{\ell=1}^{p_b} \boldsymbol{b}_\ell \, w_\ell(\boldsymbol{x}) \,. \tag{2.5.12}$$

For highly correlated output, (2.5.12) is typically a very economical representation requiring few basis functions to provide a close approximation to $\boldsymbol{y}(\boldsymbol{x})$. The coefficients $w_1(\boldsymbol{x}), \ldots, w_{p_b}(\boldsymbol{x})$ contain the information in $\boldsymbol{y}(\boldsymbol{x})$. Ideally, the basis functions are selected to be *orthogonal* so that $w_1(\boldsymbol{x}), \ldots, w_{p_b}(\boldsymbol{x})$ can be reasonably modeled as independent.

If training data are collected at $\boldsymbol{x}_1^{tr}, \ldots, \boldsymbol{x}_n^{tr}$ with associated outputs $\boldsymbol{y}(\boldsymbol{x}_1^{tr}), \ldots, \boldsymbol{y}(\boldsymbol{x}_n^{tr})$ then a prediction model $\widehat{w}_\ell(\boldsymbol{x})$ is built for each $\ell = 1, \ldots, p_b$ using the methods of Sects. 3.1–3.4 based on the "output" $w_\ell(\boldsymbol{x}_1^{tr}), \ldots, w_\ell(\boldsymbol{x}_n^{tr})$. The function $y(\boldsymbol{x}^{te})$ at the test site $\boldsymbol{x}^{te}$ is predicted by

$$\widehat{\boldsymbol{y}}(\boldsymbol{x}^{te}) = \sum_{\ell=1}^{p_b} \boldsymbol{b}_\ell \, \widehat{w}_\ell(\boldsymbol{x}^{te}).$$

Several practical issues must be resolved in order to carry out the prediction method described above: a basis system must be selected, a method of determining the coefficients $w_\ell(\boldsymbol{x})$ (usually to minimize the approximation error for a given p_b), and the number of basis vectors p_b to be used. Several methods have been used in the literature to identify basis vectors. These include the singular value decomposition applied to a standardized version of the $m \times n$ matrix:

$$\left[\boldsymbol{y}(\boldsymbol{x}_1^{tr}) \cdots \boldsymbol{y}(\boldsymbol{x}_n^{tr})\right]$$

(see Sect. 8.4.1), a wavelet basis formed from the same matrix (Bayarri et al. (2007)), and a radial basis function approximation (Bliznyuk et al. (2008)). Once the basis system has been selected as well as the method of choosing the associated $w_\ell(\boldsymbol{x})$ for a given p_b, one simple choice of p_b is the *smallest* value so that

$$\max_{1 \le i \le n} \left\| \boldsymbol{y}(\boldsymbol{x}_i^{tr}) - \sum_{\ell=1}^{p_b} \boldsymbol{b}_\ell \, \widehat{w}_\ell(\boldsymbol{x}_i^{tr}) \right\|_2 < \epsilon$$

for some given error bound $\epsilon > 0$ where $\|\bullet\|_2$ denotes Euclidian distance. Another method comes from multivariate analysis: select p_b so that the proportion of the total variability of $\boldsymbol{y}(\boldsymbol{x}_1^{tr}), \ldots, \boldsymbol{y}(\boldsymbol{x}_n^{tr})$ explained by $\sum_{\ell=1}^{p_b} \boldsymbol{b}_\ell \, w_\ell(\boldsymbol{x})$ is sufficiently large.

2.6 Chapter Notes

There are many sources that expand the overview material provided in this chapter, particularly for Gaussian processes which are introduced in Sect. 2.2. Abrahamsen (1997) is an approximately 60-page document that is available on the web; it presents an introduction to GPs, with descriptions of their sample properties and figures that provide the reader with an visual understanding of the sample paths produced by many different correlation families. To describe the smoothness of draws from a general processes is beyond the scope of this book (however, see Cramér and Leadbetter (1967), Adler (1990), Yaglom (1986)). Stein (1999) focuses on the relationship between the correlation function of a stationary GP $Y(\boldsymbol{x})$ and the smoothness properties of its realizations $y(\boldsymbol{x})$.

The earlier sections have centered their discussion on the cases of quantitative and qualitative *inputs* for real-valued, multivariate output and functional output. Particularly in cases of functional output, the inputs to simulation models can also be functional. For example, the forces exerted on a prothetic joint can vary in magnitude and angle of application over a gait cycle. Morris (2012, 2014) treat both design issues and prediction for such cases.

Several papers have proposed alternative multivariate models based on the correlation models presented Sect. 2.2. Gneiting et al. (2010) develop multivariate models for the Matérn correlation function.

Chapter 3
Empirical Best Linear Unbiased Prediction of Computer Simulator Output

3.1 Introduction

This chapter and Chap. 4 discuss techniques for predicting output for a computer simulator based on "training" runs from the model. Knowing how to predict computer output is a prerequisite for answering most practical research questions that involve computer simulators including those listed in Sect. 1.3. As an example where the prediction methods described below will be central, Chap. 6 will present a sequential design for a computer experiment to find input conditions $\boldsymbol{x}$ that maximize a computer output which requires prediction of $y(\boldsymbol{x})$ at all untried sites.

To fix ideas, this section uses "training" data $(\boldsymbol{x}_i^{tr}, y(\boldsymbol{x}_i^{tr}))$, $1 \le i \le n_s$, of the simulator to predict $y(\boldsymbol{x}^{te})$ where $\boldsymbol{x}^{te}$ is a "test" input. The $(n_s + 1)$ data are assumed to be a draw from the regression plus stationary Gaussian process model of Chap. 2 which is repeated here for convenience:

$$Y(\boldsymbol{x}) = \sum_{j=1}^{p} f_j(\boldsymbol{x})\,\beta_j + Z(\boldsymbol{x}) = \boldsymbol{f}^\top(\boldsymbol{x})\,\boldsymbol{\beta} + Z(\boldsymbol{x}), \tag{3.1.1}$$

where $\boldsymbol{f}(\boldsymbol{x})$ are known regression functions, $\boldsymbol{\beta} \in \mathbb{R}^p$ are unknown regression coefficients, and $Z(\cdot)$ is a stationary Gaussian process (GP) with zero mean, unknown process variance $\sigma_Z^2 > 0$, and known correlation function $R(\cdot)$.

At first view, the problem of determining $y(\boldsymbol{x}^{te})$ based on training data might be considered as one of *point estimation* of a *fixed population quantity*, $y(\boldsymbol{x}^{te})$. Instead, our viewpoint of regarding the entire function as drawn from a process $Y(\boldsymbol{x})$ is consistent with formulating the problem as one of *predicting* the random variable $Y(\boldsymbol{x}^{te})$ given the partial information about the draw contained in the training data $Y(\boldsymbol{x}_1^{tr}), \ldots, Y(\boldsymbol{x}_{n_s}^{tr})$.

Section 3.2 describes *two statistical approaches* to the prediction problem: best linear unbiased prediction (BLUP) and minimum mean squared prediction error (MSPE) methodology. Using slightly different versions of the model (3.1.1), both methodologies produce the same class of predictors, but the result changes with the

T. J. Santner et al., *The Design and Analysis of Computer Experiments*, Springer Series in Statistics, https://doi.org/10.1007/978-1-4939-8847-1_3

correlation structure of the GP which is typically unknown in practical problems. Section 3.3 presents empirical adaptions of the BLUP (called "EBLUP"s) which can be used in unknown correlation settings. The chapter concludes with a simulation study that compares the mean squared prediction errors of empirical best linear unbiased predictors corresponding to different correlation parameter estimators in a test bed of examples and makes recommendations.

3.2 BLUP and Minimum MSPE Predictors

3.2.1 *Best Linear Unbiased Predictors*

To explain best linear unbiased prediction in a somewhat general setting, consider a generic setup in which it is desired to predict a random variable Y_0 based on training data $\boldsymbol{Y}^{tr} = (Y_1, \ldots, Y_n)^\top$. Let $\widehat{Y_0} = \widehat{Y_0}(\boldsymbol{Y}^{tr})$ denote an arbitrary predictor of Y^{te} based on $\boldsymbol{Y}^{tr}$. The class of *linear unbiased predictors* (LUPs) of Y_0 with respect to a given family $\mathcal{F}$ of (joint) distributions for $(Y_0, \boldsymbol{Y}^{tr})$ are those $\widehat{Y_0} = a_0 + \boldsymbol{a}^\top \boldsymbol{Y}^{tr}$ which satisfy

$$E\left[\widehat{Y_0}\right] = E\left[Y_0\right], \quad \text{for all } F \in \mathcal{F}, \tag{3.2.1}$$

where $E[\cdot]$ denotes expectation under $F(\cdot)$. The condition that (3.2.1) holds is termed "unbiasedness" of $\widehat{Y_0}$ for Y^{te} with respect to $\mathcal{F}$.

This section is concerned with identifying best LUPs of Y_0 in the sense of minimizing the *mean squared prediction error* (MSPE). The MSPE of the Y_0 predictor $\widehat{Y_0}$ at F is

$$\text{MSPE}\left(\widehat{Y_0}, F\right) \equiv E\left[\left(\widehat{Y_0} - Y_0\right)^2\right]. \tag{3.2.2}$$

Definition. The LUP $\widehat{Y_0} = a_0 + \boldsymbol{a}^\top \boldsymbol{Y}^{tr}$ of Y_0 is a *best LUP* (BLUP) with respect to $\mathcal{F}$ provided

$$\text{MSPE}\left(\widehat{Y_0}, F\right) \le \text{MSPE}\left(Y_0^\star, F\right), \quad \text{for all } F \in \mathcal{F},$$

for any alternative LUP $Y_0^\star$.

Example 3.1 (LUPs for a Location Parameter Model). This simple example illustrates how the choice of $\mathcal{F}$ determines the unbiasedness of linear predictors. Suppose that $\mathcal{F}$ is specified for the data by the model statement

$$Y_i = \beta_0 + \epsilon_i$$

for $0 \le i \le n$, where β_0 is a *given nonzero* value and $\{\epsilon_i\}_{i=0}^n$ are uncorrelated zero mean random variables with unknown variance $\sigma_\epsilon^2 > 0$. This model specifies the first two moments of $(Y_0, \boldsymbol{Y}^{tr})$. An alternate statement of $\mathcal{F}$ is that $Y_0, Y_1, \ldots, Y_n$ are uncorrelated with *given* mean β_0 and unknown (positive) variance.

Letting $\boldsymbol{a} = (a_1, \ldots, a_n)^\top$, the predictor $\widehat{Y_0} = a_0 + \boldsymbol{a}^\top \boldsymbol{Y}^{tr}$ is unbiased with respect to $\mathcal{F}$ provided:

$$E\left[\widehat{Y}_0\right] = E\left[a_0 + \sum_{i=1}^{n} a_i Y_i\right] = a_0 + \beta_0 \sum_{i=1}^{n} a_i$$
$$\overset{\text{set}}{=} E\left[Y_0\right] = \beta_0 \tag{3.2.3}$$

for all $\sigma_\epsilon^2 > 0$. Because (3.2.3) is independent of σ_ϵ^2, $\widehat{Y}_0$ is unbiased provided $(a_0, \boldsymbol{a})$ satisfies

$$a_0 + \beta_0 \sum_{i=1}^{n} a_i = \beta_0 \tag{3.2.4}$$

for the given β_0. One set of $(a_0, \boldsymbol{a})$ satisfying (3.2.4) are those for which $a_0 = \beta_0$ and $\sum_{i=1}^{n} a_i = 0$; one such $\boldsymbol{a}$ is $a_1 = \cdots = a_n = 0$ which gives the (data independent) predictor $\widehat{Y}_0 = \beta_0$. Other LUPs result by choosing $a_0 = 0$ and any $\boldsymbol{a}$ for which $\sum_{i=1}^{n} a_i = 1$; for example, the sample mean of $Y_1, \ldots, Y_n$ is the LUP of Y_0 corresponding to $a_1 = \cdots = a_n = 1/n$.

Turning attention to finding the LUP of Y_0 with smallest MSPE, first observe that the MSPE of the LUP $\widehat{Y}_0 = a_0 + \boldsymbol{a}^\top \boldsymbol{Y}^n$ is

$$E\left[\left(a_0 + \sum_{i=1}^{n} a_i Y_i - Y_0\right)^2\right] = E\left[\left(a_0 + \sum_{i=1}^{n} a_i \left(\beta_0 + \epsilon_i\right) - \beta_0 - \epsilon_0\right)^2\right]$$
$$= \left(a_0 + \beta_0 \sum_{i=1}^{n} a_i - \beta_0\right)^2 + \sigma_\epsilon^2 \times \sum_{i=1}^{n} a_i^2 + \sigma_\epsilon^2$$
$$= \sigma_\epsilon^2 \times \left(1 + \sum_{i=1}^{n} a_i^2\right) \tag{3.2.5}$$
$$\geq \sigma_\epsilon^2 \,. \tag{3.2.6}$$

Equality holds in (3.2.5) because $\widehat{Y}_0$ is unbiased, and equality occurs in (3.2.6) if and only if $a_0 = \beta_0$ and $a_1 = \cdots = a_n = 0$, which shows that

$$\widehat{Y}_0 = \beta_0$$

is the *unique BLUP* for $\mathcal{F}$. For this example, as is often the case for BLUPs, $\widehat{Y}_0$ depends heavily on $\mathcal{F}$ and, indeed, changes with each different β_0. ♦

Returning to the problem that motivates this chapter, consider predicting $y(\boldsymbol{x}^{te})$ based on $(\boldsymbol{x}_i^{tr}, y(\boldsymbol{x}_i^{tr}))$, $1 \leq i \leq n_s$, where the data satisfy model (3.1.1) and the correlation function $R(\cdot)$ is *known*. Using a slight change of notation in which the predictor is denoted by $\widehat{y}(\boldsymbol{x}^{te})$ rather than $\widehat{Y}(\boldsymbol{x}^{te})$ to be more suggestive of this application, Sect. 3.6.1 shows that

$$\boxed{\widehat{y}(\boldsymbol{x}^{te}) = \widehat{y}^{te} \equiv \boldsymbol{f}_{te}^{\top}\widehat{\boldsymbol{\beta}} + \boldsymbol{r}_{te}^{\top}\boldsymbol{R}_{tr}^{-1}(\boldsymbol{Y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}) \qquad (3.2.7)}$$

is the BLUP of $Y(\boldsymbol{x}^{te})$ with respect to the family of distributions (3.1.1). In (3.2.7):

- $\boldsymbol{R}_{tr} = \left(R(\boldsymbol{x}_i^{tr} - \boldsymbol{x}_j^{tr})\right)$ is the $n_s \times n_s$ matrix of *known* correlations of model outputs at all pairs of training data inputs;
- $\boldsymbol{F}_{tr}$ is the $n_s \times p$ *known* matrix whose i^{th} row is $\left(f_1(\boldsymbol{x}_i^{tr}), \ldots, f_p(\boldsymbol{x}_i^{tr})\right)$, $1 \le i \le n_s$;
- $\widehat{\boldsymbol{\beta}} = \left(\boldsymbol{F}_{tr}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr}\right)^{-1} \boldsymbol{F}_{tr}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{Y}^{tr}$ is the generalized least squares estimator of $\boldsymbol{\beta}$;
- $\boldsymbol{f}_{te}^{\top} = \left(f_1(\boldsymbol{x}^{te}), \ldots, f_p(\boldsymbol{x}^{te})\right)$ is the $1 \times p$ vector of *known* regressors at the test data input;
- $\boldsymbol{r}_{te}^{\top} = \left(R(\boldsymbol{x}^{te} - \boldsymbol{x}_1^{tr}), \ldots, R(\boldsymbol{x}^{te} - \boldsymbol{x}_{n_s}^{tr})\right)$ is the $1 \times n_s$ vector of *known* correlations of $Y(\boldsymbol{x}^{te})$ with each training data model output $Y(\boldsymbol{x}_i^{tr})$.

The uncertainty in the predictor $\widehat{y}(\boldsymbol{x}^{te})$ can be quantified by its root MSPE, i.e., the square root of

$$\text{MSPE} = s^2(\boldsymbol{x}^{te}) = E\left[\left(Y(\boldsymbol{x}^{te}) - \widehat{y}(\boldsymbol{x}^{te})\right)^2\right].$$

The expectation is with respect to the distribution of $(Y(\boldsymbol{x}^{te}), \boldsymbol{Y}^{tr})$ (from (3.2.7), $\widehat{y}(\boldsymbol{x}^{te})$ is a function of $\boldsymbol{Y}^{tr}$). In Sect. 3.6.1, the MSPE of $\widehat{y}(\boldsymbol{x}^{te})$ is derived to be

$$s^2(\boldsymbol{x}^{te}) = \sigma_Z^2\left\{1 - \boldsymbol{r}_{te}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{r}_{te} + \boldsymbol{h}^{\top}\boldsymbol{Q}^{-1}\boldsymbol{h}\right\}, \qquad (3.2.8)$$

where $\boldsymbol{h} = \boldsymbol{f}_{te} - \boldsymbol{F}_{tr}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{r}_{te}$ and $\boldsymbol{Q} = \boldsymbol{F}_{tr}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr}$. The uncertainty quantification (3.2.8) is known only up to the scalar σ_Z^2 for the model discussed in this subsection, where it has been assumed that $\boldsymbol{\beta} \in \mathbb{R}^p$ and $\sigma_Z^2 > 0$, while both $\boldsymbol{r}_{te}$ and $\boldsymbol{R}_{tr}$ are known.

In sum, because the BLUP (3.2.7) depends on the correlation structure, a quantity that is not known in detail in most practical applications, the formula (3.2.7) is of limited direct application. However both (3.2.7) and (3.2.8) are of use in applied work where estimated correlation functions and process variance are used in place of known ones.

The next subsection will show that (3.2.7) also arises as the result of invoking a second classical statistical principle for identifying optimal predictors, that of minimum MSPE prediction; this methodology allows predictors of *arbitrary form* but considers a *more restricted process model* than in the derivation of the BLUP.

3.2.2 Best MSPE Predictors

Again, initially consider the generic case in which it is desired to predict Y_0 based on training data $\boldsymbol{Y}^{tr} = (Y_1, \ldots, Y_n)^{\top}$.

Definition. The predictor $\widehat{Y}_0$ of Y_0 is a *minimum (best) MSPE predictor* at F provided

$$\text{MSPE}\left(\widehat{Y}_0, F\right) \le \text{MSPE}\left(Y_0^{\star}, F\right)$$

for *any* alternative predictor $Y_0^{\star}$ where (3.2.2) defines the MSPE.

Predictors of practical importance will simultaneously minimize the MSPE for *many* distributions F.

Theorem 3.1 is the fundamental theorem of prediction; it states that the conditional mean of Y_0 given $\boldsymbol{Y}^{tr}$ is the minimum MSPE predictor of Y_0 based on $\boldsymbol{Y}^{tr}$ (provided the moments referred to below exist). The expectation notation refers to the distributions of $(Y_0, \boldsymbol{Y}^{tr})$ induced by F.

Theorem 3.1. Suppose that $(Y_0, \boldsymbol{Y}^{tr})$ have distribution F. Then

$$\widehat{Y}_0 = E\left[Y_0 \,\middle|\, \boldsymbol{Y}^{tr}\right] \tag{3.2.9}$$

is the best MSPE predictor of Y_0.

Proof: Choose an arbitrary competing predictor $Y_0^{\star} = Y_0^{\star}(\boldsymbol{Y}^{tr})$; then

$$\begin{aligned}
\text{MSPE}\left(Y_0^{\star}, F\right) &= E\left[(Y_0^{\star} - Y_0)^2\right] \\
&= E\left[\left(Y_0^{\star} - \widehat{Y}_0 + \widehat{Y}_0 - Y_0\right)^2\right] \\
&= E\left[\left(Y_0^{\star} - \widehat{Y}_0\right)^2\right] + \text{MSPE}\left(\widehat{Y}_0, F\right) \\
&\quad + 2E\left[\left(Y_0^{\star} - \widehat{Y}_0\right)\left(\widehat{Y}_0 - Y_0\right)\right] \\
&\ge \text{MSPE}\left(\widehat{Y}_0, F\right) + 2E\left[\left(Y_0^{\star} - \widehat{Y}_0\right)\left(\widehat{Y}_0 - Y_0\right)\right] && (3.2.10) \\
&= \text{MSPE}\left(\widehat{Y}_0, F\right), && (3.2.11)
\end{aligned}$$

where (3.2.10) holds because $E[(Y_0^{\star} - \widehat{Y}_0)^2] \ge 0$. Equality holds in (3.2.11) because

$$\begin{aligned}
E\left[\left(Y_0^{\star} - \widehat{Y}_0\right)\left(\widehat{Y}_0 - Y_0\right)\right] &= E\left[\left(Y_0^{\star} - \widehat{Y}_0\right) E\left[\left(\widehat{Y}_0 - Y_0\right) \middle| \, \boldsymbol{Y}^{tr}\right]\right] \\
&= E\left[\left(Y_0^{\star} - \widehat{Y}_0\right)\left(\widehat{Y}_0 - E\left[Y_0 \,\middle|\, \boldsymbol{Y}^{tr}\right]\right)\right] \\
&= E\left[\left(Y_0^{\star} - \widehat{Y}_0\right) \times 0\right] \\
&= 0\,,
\end{aligned}$$

completing the proof. ■

Best MSPE predictors must be *unbiased* with respect to any F for which (3.2.9) holds because

$$E\left[\widehat{Y}_0\right] = E\left[E\left[Y_0 \,\middle|\, \boldsymbol{Y}^{tr}\right]\right] = E\left[Y_0\right].$$

Hence, best MSPE predictors represent a strengthening of the BLUP criterion in that the class of predictors considered are not merely those linear in the training data but can be arbitrary. The next example shows that improvements in MSPE are possible by increasing the class of predictors beyond linear ones.

Example 3.2. Suppose that (Y_0, Y_1) has the joint distribution given by the density

$$f(y_0, y_1) = \begin{cases} 1/y_1^2\,, \; 0 < y_1 < 1, 0 < y_0 < y_1^2 \\ 0\,, \quad \text{otherwise.} \end{cases}$$

It is straightforward to calculate that the conditional distribution of Y_0 given $Y_1 = y_1$ is uniform over the interval $(0, y_1^2)$. Hence the best MSPE predictor of Y_0 is the center of this interval, i.e.,

$$\widehat{Y}_0 = E\,[Y_0 \,|\, Y_1] = Y_1^2/2$$

which is nonlinear in Y_1.

We compare the best MSPE predictor with the BLUP of Y_0 for this same $f(y_0, y_1)$ model for (Y_0, Y_1). The BLUP is that $a_0 + a_1 Y_1$ which minimizes $E[(a_0 + a_1 Y_1 - Y_0)^2]$ among those (a_0, a_1) that satisfy the unbiasedness requirement $E[a_0 + a_1 Y_1] = E[Y_0]$. Unbiasedness leads to the requirement

$$a_0 + a_1 \frac{1}{2} = \frac{1}{6} \quad \text{or} \quad a_0 = \frac{1}{6} - a_1 \frac{1}{2}\,.$$

Applying calculus to minimize the MSPE

$$E\left[\left(\left(\frac{1}{6} - a_1 \frac{1}{2}\right) + a_1 Y_1 - Y_0\right)^2\right]$$

shows that $a_1 = 1/2$ and hence $a_0 = 1/6 - a_1/2 = -1/12$, i.e., $\widehat{Y}_0^L = -\frac{1}{12} + \frac{1}{2} Y_1$ is the BLUP of Y_0.

Figure 3.1 shows that the predictors $\widehat{Y}_0$ and $\widehat{Y}_0^L$ are very close for all $y_1 \in (0, 1)$. The MSPE of $\widehat{Y}_0$ is

$$\begin{aligned} E\left[\left(Y_0 - Y_1^2/2\right)^2\right] &= E\left[E\left[(Y_0 - Y_1^2/2)^2 \,\middle|\, Y_1\right]\right] \\ &= E\,[Var\,[Y_0 \,|\, Y_1]] \\ &= E\left[Y_1^4/12\right] \qquad (3.2.12) \\ &= 1/60 \approxeq 0.01667\,. \end{aligned}$$

The inner term in (3.2.12), $Y_1^4/12$, is the variance of the uniform distribution over $(0, y_1^2)$. A similar calculation gives

$$E\left[\left(\widehat{Y}_0^L - Y_0\right)^2\right] = 0.01806 > 0.01667$$

as theory dictates, but the difference is small, as Fig. 3.1 suggests. ♦

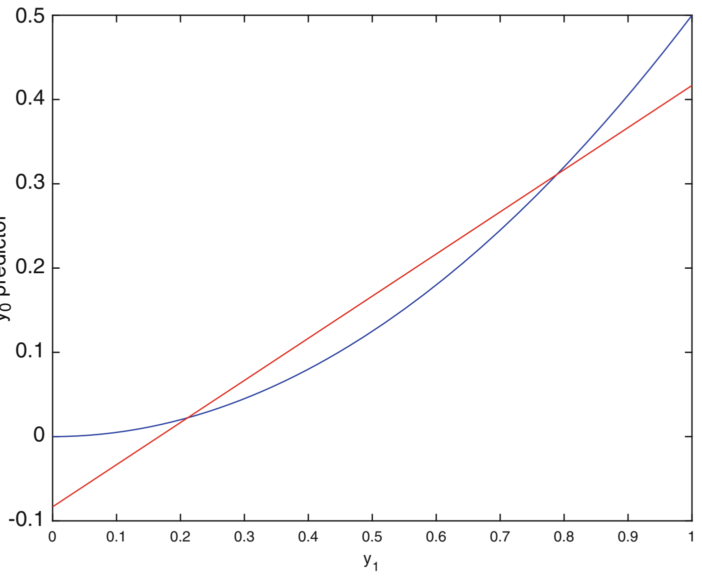


Fig. 3.1 For example 3.2, the predictors $\widehat{Y}_0$ (solid blue) and $\widehat{Y}_0^L$ (solid red) versus $y_1 \in (0, 1)$

Example 3.3. Returning to the GP setting with *known* correlation, momentarily assume that both $\boldsymbol{\beta}$ and σ_Z^2 are also *known*. Then the joint distribution of $Y^{te} = Y(\boldsymbol{x}^{te})$ and $\boldsymbol{Y}^{tr} = (Y(\boldsymbol{x}_1^{tr}), \dots, Y(\boldsymbol{x}_{n_s}^{tr}))^\top$ (given $\boldsymbol{\beta}$, σ_Z^2, and $R(\cdot)$) is multivariate normal:

$$\begin{pmatrix} Y^{te} \\ \boldsymbol{Y}^{tr} \end{pmatrix} \sim N_{1+n_s}\left(\begin{pmatrix} \boldsymbol{f}_{te}^\top \\ \boldsymbol{F}_{tr} \end{pmatrix} \boldsymbol{\beta},\ \sigma_Z^2 \begin{bmatrix} 1 & \boldsymbol{r}_{te}^\top \\ \boldsymbol{r}_{te} & \boldsymbol{R}_{tr} \end{bmatrix} \right). \tag{3.2.13}$$

Now, assuming that the design matrix $\boldsymbol{F}_{tr}$ is of full column rank p and that $\boldsymbol{R}_{tr}$ is positive definite, Theorems 3.1 and B.2 show that

$$\widehat{y}^{te} = E\left[Y^{te} \,\middle|\, \boldsymbol{Y}^{tr} \right] = \boldsymbol{f}_{te}^\top \boldsymbol{\beta} + \boldsymbol{r}_{te}^\top \boldsymbol{R}_{tr}^{-1} \left(\boldsymbol{Y}^{tr} - \boldsymbol{F}_{tr} \boldsymbol{\beta} \right) \tag{3.2.14}$$

is the best MSPE predictor of $y(\boldsymbol{x}^{te})$.

The class of distributions $\mathcal{F}$ for which (3.2.14) is the minimum MSPE predictor is, again, embarrassingly small. The best MSPE predictor is altered when either $\boldsymbol{\beta}$ or $R(\cdot)$ changes; however, $\widehat{y}^{te}$ is the same for all $\sigma_Z^2 > 0$. ♦

Example 3.4. Consider a second version of the regression plus stationary GP model in which $\boldsymbol{\beta}$ is *unknown* and σ_Z^2 is *known* (although this is not needed in the calculation below). Assume that the data follow a two-stage model specified as follows.

View (3.2.13) as specifying the conditional distribution of $(Y^{te}, \boldsymbol{Y}^{tr})$ given $\boldsymbol{\beta}$, i.e., $[(Y^{te}, \boldsymbol{Y}^{tr}) \,|\, \boldsymbol{\beta}]$ is the first stage of a two-stage model. The second stage of the model puts an arbitrary prior on $\boldsymbol{\beta}$, which will be denoted by $[\boldsymbol{\beta}]$. The best MSPE predictor of Y^{te} is

$$\widehat{y}^{te} = E\left[Y^{te} \mid \boldsymbol{Y}^{tr}\right] = E\left[E\left[Y^{te} \mid \boldsymbol{Y}^{tr}, \boldsymbol{\beta}\right] \Big| \, \boldsymbol{Y}^{tr}\right]$$
$$= E\left[\boldsymbol{f}_{te}^{\top}\boldsymbol{\beta} + \boldsymbol{r}_{te}^{\top}\boldsymbol{R}_{tr}^{-1}\left(\boldsymbol{Y}^{tr} - \boldsymbol{F}_{tr}\boldsymbol{\beta}\right) \Big| \, \boldsymbol{Y}^{tr}\right]$$

and the last expectation is with respect to the conditional distribution of $\boldsymbol{\beta}$ given $\boldsymbol{Y}^{tr}$. Thus

$$\widehat{y}^{te} = \boldsymbol{f}_{te}^{\top}\, E\left[\boldsymbol{\beta} \mid \boldsymbol{Y}^{tr}\right] + \boldsymbol{r}_{te}^{\top}\, \boldsymbol{R}_{tr}^{-1}\left(\boldsymbol{Y}^{tr} - \boldsymbol{F}_{tr}\, E\left[\boldsymbol{\beta} \mid \boldsymbol{Y}^{tr}\right]\right)$$

is the minimum MSPE predictor of $Y(\boldsymbol{x}^{te})$ for *any* two-stage model whose first stage is given by (3.2.13) and has arbitrary second stage $\boldsymbol{\beta}$ prior for which $E[\boldsymbol{\beta} \mid \boldsymbol{Y}^{tr}]$ exists.

Of course, the explicit formula for $E[\boldsymbol{\beta} \mid \boldsymbol{Y}^{tr}]$, and hence $\widehat{y}^{te}$, depends on the $\boldsymbol{\beta}$ prior. For example, when $\boldsymbol{\beta}$ has the non-informative prior

$$[\,\boldsymbol{\beta}\,] \propto 1,$$

the conditional distribution $[\boldsymbol{\beta} \mid \boldsymbol{Y}^{tr}]$ can be derived by observing

$$\left[\boldsymbol{\beta} \mid \boldsymbol{Y}^{tr} = \boldsymbol{y}^{tr}\right] \propto \left[\boldsymbol{y}^{tr} \mid \boldsymbol{\beta}\right] \times [\boldsymbol{\beta}]$$
$$\propto \exp\left\{-\frac{1}{2\sigma_Z^2}\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\boldsymbol{\beta}\right)^{\top}\boldsymbol{R}_{tr}^{-1}\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\boldsymbol{\beta}\right)\right\} \times 1$$
$$\propto \exp\left\{-\frac{1}{2\sigma_Z^2}\left(\boldsymbol{\beta}^{\top}\boldsymbol{F}_{tr}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr}\boldsymbol{\beta} - 2\boldsymbol{\beta}^{\top}\boldsymbol{F}_{tr}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{y}^{tr}\right)\right\}$$
$$= \exp\left\{-\frac{1}{2}\boldsymbol{\beta}^{\top}\boldsymbol{A}^{-1}\boldsymbol{\beta} + \boldsymbol{\nu}^{\top}\boldsymbol{\beta}\right\},$$

where $\boldsymbol{A}^{-1} = \boldsymbol{F}_{tr}^{\top}(\sigma_Z^2\boldsymbol{R}_{tr})^{-1}\boldsymbol{F}_{tr}$ and $\boldsymbol{\nu} = \boldsymbol{F}_{tr}^{\top}(\sigma_Z^2\boldsymbol{R}_{tr})^{-1}\boldsymbol{y}^{tr}$. Notice that rank$(\boldsymbol{A}) = p$ under the continuing assumption that $\boldsymbol{F}$ has full column rank p. Applying (B.1.4) of Appendix B gives

$$\left[\boldsymbol{\beta} \mid \boldsymbol{Y}^{tr}\right] \sim N_p\left((\boldsymbol{F}_{tr}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr})^{-1}\boldsymbol{F}_{tr}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{Y}^{tr},\; \sigma_Z^2\,(\boldsymbol{F}_{tr}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr})^{-1}\right)$$

because the σ_Z^2 terms cancel in the expression for the mean of $[\boldsymbol{\beta} \mid \boldsymbol{Y}^{tr}]$. Thus the best MSPE predictor of Y_0 under this two-stage model is

$$\widehat{y}^{te} = \boldsymbol{f}_{te}^{\top}\widehat{\boldsymbol{\beta}} + \boldsymbol{r}_{te}^{\top}\boldsymbol{R}_{tr}^{-1}\left(\boldsymbol{Y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right), \tag{3.2.15}$$

where $\widehat{\boldsymbol{\beta}} = (\boldsymbol{F}_{tr}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr})^{-1}\boldsymbol{F}_{tr}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{Y}^{tr}$ which is the BLUP as stated in Sect. 3.2.1. ♦

3.2.3 Some Properties of $\widehat{y}(\boldsymbol{x}^{te})$

We conclude this discussion of the BLUP (3.2.7) by describing several of its properties that are straightforward to demonstrate.

- *The BLUP is linear in the training data $\boldsymbol{Y}^{tr}$*: linearity follows by substituting $\widehat{\boldsymbol{\beta}}$ into $\widehat{y}^{te}$ yielding

$$\begin{aligned}\widehat{y}^{te} &= \Big[\boldsymbol{f}_{te}^{\top}(\boldsymbol{F}_{tr}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr})^{-1}\boldsymbol{F}_{tr}^{\top}\boldsymbol{R}_{tr}^{-1} \\ &\quad + \boldsymbol{r}_{te}^{\top}\boldsymbol{R}_{tr}^{-1}(\boldsymbol{I}_{n_s} - \boldsymbol{F}_{tr}(\boldsymbol{F}_{tr}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr})^{-1}\boldsymbol{F}_{tr}^{\top}\boldsymbol{R}_{tr}^{-1})\Big]\boldsymbol{Y}^{tr} \\ &\equiv \boldsymbol{a}_{*}^{\top}\boldsymbol{Y}^{tr},\end{aligned} \tag{3.2.16}$$

  where (3.2.16) defines $\boldsymbol{a}_*$.
- *The BLUP is unbiased with respect to (3.1.1)*: for any $\boldsymbol{\beta} \in \mathbb{R}^p$ and every $\sigma_Z^2 > 0$,

$$\begin{aligned}E\left[\widehat{y}^{te}\right] &= \boldsymbol{a}_{*}^{\top}E\left[\boldsymbol{Y}^{tr}\right] = \boldsymbol{a}_{*}^{\top}\boldsymbol{F}_{tr}\boldsymbol{\beta} \\ &= \left[\boldsymbol{f}_{te}^{\top}\boldsymbol{I}_p + \boldsymbol{r}_{te}^{\top}\boldsymbol{R}_{tr}^{-1}(\boldsymbol{F}_{tr} - \boldsymbol{F}_{tr}\boldsymbol{I}_p)\right]\boldsymbol{\beta} \\ &= \boldsymbol{f}_{te}^{\top}\boldsymbol{\beta} = E\left[Y(\boldsymbol{x}^{te})\right],\end{aligned} \tag{3.2.17}$$

  where (3.2.17) holds from algebra after substituting for $\boldsymbol{a}_*$ from (3.2.16). As shown in Sect. 3.6.1, the unbiasedness of (3.2.7) is the key fact that permits the straightforward derivation of its variance optimality.
- *The BLUP interpolates the training data $(\boldsymbol{x}_i^{tr}, Y(\boldsymbol{x}_i^{tr}))$, $1 \le i \le n_s$.* First, notice that $\widehat{y}^{te}$ can be regarded as the sum of the regression predictor $\boldsymbol{f}_{te}^{\top}\widehat{\boldsymbol{\beta}}$ plus the "correction" $\boldsymbol{r}_{te}^{\top}\boldsymbol{R}_{tr}^{-1}\left(\boldsymbol{Y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right)$. Suppose that $\boldsymbol{x}^{te} = \boldsymbol{x}_i^{tr}$ for some fixed i, $1 \le i \le n_s$. Then $\boldsymbol{f}_{te} = \boldsymbol{f}^{\top}(\boldsymbol{x}_i^{tr})$ and

$$\boldsymbol{r}_{te}^{\top} = \left(R(\boldsymbol{x}_i^{tr} - \boldsymbol{x}_1^{tr}), R(\boldsymbol{x}_i^{tr} - \boldsymbol{x}_2^{tr}), \ldots, R(\boldsymbol{x}_i^{tr} - \boldsymbol{x}_{n_s}^{tr})\right)$$

  which is the ith row of $\boldsymbol{R}_{tr}$. Thus $\boldsymbol{R}_{tr}^{-1}\boldsymbol{r}_{te} = (0, \ldots, 0, 1, 0, \ldots, 0)^{\top} = \boldsymbol{e}_i$, the ith unit vector because this product is the ith column of $\boldsymbol{R}_{tr}^{-1}\boldsymbol{R}_{tr} = \boldsymbol{I}_{n_s}$, the $n_s \times n_s$ identity matrix. Hence

$$\boldsymbol{r}_{te}^{\top}\boldsymbol{R}_{tr}^{-1}\left(\boldsymbol{Y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right) = \boldsymbol{e}_i^{\top}\left(\boldsymbol{Y}^{tr} - \boldsymbol{F}_{tr}\,\widehat{\boldsymbol{\beta}}\right) = Y(\boldsymbol{x}_i^{tr}) - \boldsymbol{f}^{\top}(\boldsymbol{x}_i^{tr})\widehat{\boldsymbol{\beta}}$$

  and so

$$\widehat{y}(\boldsymbol{x}^{te}) = \boldsymbol{f}^{\top}(\boldsymbol{x}_i^{tr})\,\widehat{\boldsymbol{\beta}} + \left(Y(\boldsymbol{x}_i^{tr}) - \boldsymbol{f}^{\top}(\boldsymbol{x}_i^{tr})\,\widehat{\boldsymbol{\beta}}\right) = Y(\boldsymbol{x}_i^{tr}).$$

- *Regarded as a function of $\boldsymbol{x}^{te}$, the BLUP is a linear combination of the "basis" functions $\left\{R(\boldsymbol{x}_i^{tr} - \boldsymbol{x}^{te})\right\}_{i=1}^{n_s}$ and the regressors $\left\{f_j(\boldsymbol{x}^{te})\right\}_{j=1}^{p}$* From (3.2.7), the BLUP depends on $\boldsymbol{x}^{te}$ only through the vector $\boldsymbol{r}_{te} = (R(\boldsymbol{x}^{te} - \boldsymbol{x}_1^{tr}), \ldots, R(\boldsymbol{x}^{te} - \boldsymbol{x}_{n_s}^{tr}))^{\top}$ and the regression coefficients $\boldsymbol{f}(\boldsymbol{x}^{te})$,

$$\widehat{y}(\boldsymbol{x}^{te}) = \sum_{j=1}^{p} \widehat{\beta}_j\, f_j(\boldsymbol{x}^{te}) + \sum_{i=1}^{n_s} d_i\, R\left(\boldsymbol{x}^{te} - \boldsymbol{x}_i^{tr}\right), \tag{3.2.18}$$

where $\boldsymbol{d} = (d_1, \ldots, d_n)^\top = \boldsymbol{R}_{tr}^{-1}(\boldsymbol{Y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}})$. In the special case $Y(\boldsymbol{x}) = \beta_0 + Z(\boldsymbol{x})$, $\widehat{y}(\boldsymbol{x}^{te})$ depends on $\boldsymbol{x}^{te}$ only through the terms $R(\boldsymbol{x}^{te} - \boldsymbol{x}_i^{tr})$. The "smoothness" characteristics of $\widehat{y}(\boldsymbol{x}^{te})$ are inherited from those of $R(\cdot)$. For $\boldsymbol{x}^{te}$ "near" any $\boldsymbol{x}_i^{tr}$ (more precisely, in the limit as $\boldsymbol{x}^{te}$ approaches $\boldsymbol{x}_i^{tr}$), the behavior of $\widehat{y}(\boldsymbol{x}^{te})$ depends on that of $R(\cdot)$ at the origin.

For cases where the correlation function is known, this section has introduced the predictor (3.2.7) that is justified using two classical criteria: best linear unbiased prediction and minimum MSPE prediction. Unfortunately, this predictor is optimal within a very restricted class of competing predictors. The next section will turn to the problem of prediction for computer experiments in which the correlation structure is *unknown*.

3.3 Empirical Best Linear Unbiased Prediction of Univariate Simulator Output

3.3.1 Introduction

Suppose it is desired to predict $y(\boldsymbol{x})$ at the "test" input $\boldsymbol{x} = \boldsymbol{x}^{te}$ based on training data $(\boldsymbol{x}_i^{tr}, y(\boldsymbol{x}_i^{tr}))$, $1 \leq i \leq n_s$, assuming that $y(\boldsymbol{x})$ can be modeled as a draw from the regression plus stationary Gaussian process model (3.1.1) which is repeated here,

$$Y(\boldsymbol{x}) = \sum_{j=1}^{p} f_j(\boldsymbol{x})\, \beta_j + Z(\boldsymbol{x}) = \boldsymbol{f}^\top(\boldsymbol{x})\, \boldsymbol{\beta} + Z(\boldsymbol{x}), \tag{3.3.1}$$

for convenience. Throughout it is assumed that the correlation function is parametric with unknown parameters, i.e., $R(\cdot) = R(\cdot \mid \boldsymbol{\kappa})$ for unknown correlation parameter $\boldsymbol{\kappa}$. Also assume that the model regression parameters $\boldsymbol{\beta}$ are unknown as is the $Z(\boldsymbol{x})$ process variance σ_Z^2. For example, the power exponential correlation function

$$R(\boldsymbol{h}) = \exp\left\{-\sum_{j=1}^{d} \xi_j\, |h_j|^{p_j}\right\}$$

has d unknown rate ("length") parameters $\xi_1, \ldots, \xi_d$ and d unknown power parameters, $p_1, \ldots, p_d$. Then $\boldsymbol{\kappa} \equiv (\xi_1, \ldots, \xi_d, p_1, \ldots, p_d)$ consists of $2 \times d$ unknown parameters which are required to specify the correlation function.

The basic strategy of empirical best linear unbiased prediction (EBLUP) of $y(\boldsymbol{x}^{te})$ is to apply the BLUP (3.2.7) with estimated correlation parameters, i.e.,

$$\widehat{y}(\boldsymbol{x}^{te}) = \widehat{y}^{te} \equiv \boldsymbol{f}_{te}^{\top}\widehat{\boldsymbol{\beta}} + \widehat{\boldsymbol{r}}_{te}^{\top}\widehat{\boldsymbol{R}}_{tr}^{-1}\left(\boldsymbol{Y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right), \tag{3.3.2}$$

where $\widehat{\boldsymbol{R}}_{tr} = \left(R\left(\boldsymbol{x}_i^{tr} - \boldsymbol{x}_j^{tr} \,|\widehat{\boldsymbol{\kappa}}\right)\right)$ is $n_s \times n_s$, $\widehat{\boldsymbol{r}}_{te} = \left(R\left(\boldsymbol{x}^{te} - \boldsymbol{x}_i^{tr} \,|\widehat{\boldsymbol{\kappa}}\right)\right)$ is $n_s \times 1$, $\widehat{\boldsymbol{\kappa}}$ is an estimator of $\boldsymbol{\kappa}$, and $\widehat{\boldsymbol{\beta}} = (\boldsymbol{F}_{tr}^{\top}\widehat{\boldsymbol{R}}_{tr}^{-1}\boldsymbol{F}_{tr})^{-1}\boldsymbol{F}_{tr}^{\top}\widehat{\boldsymbol{R}}_{tr}^{-1}\boldsymbol{Y}^{tr}$. Several $\widehat{\boldsymbol{\kappa}}$ will be described in Sects. 3.3.2–3.3.5, and an example illustrating an EBLUP will be given in Sect. 3.3.6. EBLUPs corresponding to specific methods of estimating $\boldsymbol{\kappa}$ will have names prefixed by that estimation method, e.g., MLE-EBLUP and REML-EBLUP. The reader should note that EBLUPs are not linear in the training data $\boldsymbol{Y}^{tr}$ because $\widehat{\boldsymbol{\kappa}}$ is (typically) not linear in $\boldsymbol{Y}^{tr}$ and hence neither are $\widehat{\boldsymbol{R}}_{tr}$ or $\widehat{\boldsymbol{r}}_{te}$; also $\widehat{y}(\boldsymbol{x}^{te})$ need not be an unbiased predictor for $y(\boldsymbol{x}^{te})$ (although see Kackar and Harville (1984)).

This section describes four methods of estimating $\boldsymbol{\kappa}$ that have been proposed in the literature; use of (3.3.2) leads to different EBLUPs. All except the "cross-validation" estimator of $\boldsymbol{\kappa}$ require that the training data satisfy

$$\left[\boldsymbol{Y}^{tr} \,\middle|\, \boldsymbol{\beta}, \sigma_Z^2, \boldsymbol{\kappa}\right] \sim N_{n_s}\left(\boldsymbol{F}_{tr}\boldsymbol{\beta}, \sigma_Z^2\boldsymbol{R}_{tr}\right).$$

Also note that while the predictor (3.3.2) does not require knowledge of σ_Z^2, it will be shown in Chap. 4, e.g., (4.2.8), that an estimate of σ_Z^2 *is required* to quantify the uncertainty of the predictor at the new test input $\boldsymbol{x}^{te}$. Specific methods of estimating σ_Z^2 will be provided in the subsections on estimating unknown correlation parameters $\boldsymbol{\kappa}$.

3.3.2 Maximum Likelihood EBLUPs

Arguably maximum likelihood estimation (MLE) is the most popular method of estimating $\boldsymbol{\kappa}$. Using the multivariate normal assumption for $\boldsymbol{Y}^{tr}$, the log likelihood is, up to an additive constant,

$$\begin{aligned}\ell(\boldsymbol{\beta}, \sigma_Z^2, \boldsymbol{\kappa}) = -\frac{1}{2}\Big[&n_s\, \ell n\left(\sigma_Z^2\right) + \ell n\,(\det(\boldsymbol{R}_{tr})) \\ &+ \left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\boldsymbol{\beta}\right)^{\top}\boldsymbol{R}_{tr}^{-1}(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\boldsymbol{\beta})/\sigma_Z^2\Big],\end{aligned} \tag{3.3.3}$$

where $\det(\boldsymbol{R}_{tr})$ denotes the determinant of $\boldsymbol{R}_{tr}$. Given $\boldsymbol{\kappa}$, the MLE of $\boldsymbol{\beta}$ is its generalized least squares estimate

$$\widehat{\boldsymbol{\beta}} = \left(\boldsymbol{F}_{tr}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr}\right)^{-1}\boldsymbol{F}_{tr}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{y}^{tr} \tag{3.3.4}$$

and the MLE of σ_Z^2 is

$$\widehat{\sigma_Z^2} = \frac{1}{n_s}\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right)^{\top}\boldsymbol{R}_{tr}^{-1}\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right). \tag{3.3.5}$$

Substituting (3.3.4) and (3.3.5) into (3.3.3) shows that the maximum of (3.3.3) over $\boldsymbol{\beta}$ and σ_Z^2 is

$$\ell(\widehat{\boldsymbol{\beta}}, \widehat{\sigma_Z^2}, \boldsymbol{\kappa}) = -\frac{1}{2}\left[n_s\, \ell n\left(\widehat{\sigma_Z^2}(\boldsymbol{\kappa})\right) + \ell n\left(\det\left(\boldsymbol{R}_{tr}\left(\boldsymbol{\kappa}\right)\right)\right) + n_s \right],$$

where the dependence of σ_Z^2 and $\boldsymbol{R}_{tr}$ on $\boldsymbol{\kappa}$ is explicitly shown. Thus the MLE chooses $\widehat{\boldsymbol{\kappa}}$ to minimize

$$n_s\, \ell n\left(\widehat{\sigma_Z^2}(\boldsymbol{\kappa})\right) + \ell n\left(\det\left(\boldsymbol{R}_{tr}\left(\boldsymbol{\kappa}\right)\right)\right), \tag{3.3.6}$$

where $\widehat{\sigma_Z^2}$ is defined by (3.3.5). The predictor of $y(\boldsymbol{x}^{te})$ corresponding to $\widehat{\boldsymbol{\kappa}}$ is denoted as the MLE-EBLUP of $y(\boldsymbol{x}^{te})$.

3.3.3 Restricted Maximum Likelihood EBLUPs

Again assume that $R(\cdot)$ (and hence $\boldsymbol{R}_{tr}$ and $\boldsymbol{r}_{te}$) depends on an unknown finite vector of parameters $\boldsymbol{\kappa}$. Restricted ("residual") maximum likelihood estimation (REML) of variance and covariance parameters was introduced by Patterson and Thompson (1971) as a method of determining less biased estimates of such parameters than maximum likelihood estimation (see also Patterson and Thompson (1974)). Some authors use the term *marginal maximum likelihood estimates* for the same concept.

The REML estimator of $\boldsymbol{\kappa}$ maximizes the likelihood of a maximal set of linearly independent combinations of the $\boldsymbol{Y}^{tr}$ where each linear combination is orthogonal to $\boldsymbol{F}_{tr}\boldsymbol{\beta}$, the mean vector of $\boldsymbol{Y}^{tr}$. Assuming that $\boldsymbol{F}_{tr}$ is of full column rank p, this method first chooses an $(n_s - p) \times n_s$ matrix $\boldsymbol{C}$ of full row rank, $n_s - p$, that satisfies $\boldsymbol{C}\boldsymbol{F}_{tr} = \boldsymbol{0}$. The REML estimator of $\boldsymbol{\kappa}$ is the maximizer of the likelihood of the linearly transformed "data":

$$\boldsymbol{W} \equiv \boldsymbol{C}\boldsymbol{Y}^{tr} \sim N\left(\boldsymbol{C}\boldsymbol{F}_{tr}\boldsymbol{\beta} = \boldsymbol{0}\,,\ \sigma_Z^2\, \boldsymbol{C}\, \boldsymbol{R}_{tr}(\boldsymbol{\kappa})\, \boldsymbol{C}^\top\right).$$

Notice $\boldsymbol{W}$ has the disadvantage that it contains p fewer "observations" than $\boldsymbol{Y}^{tr}$, but it has the advantage that the distribution of $\boldsymbol{W}$ does not depend on the unknown $\boldsymbol{\beta}$.

Example 3.5. As an example, consider the simplest linear model setting, that of independent and identically distributed $N(\beta_0, \sigma^2)$ observations $Y_1, \ldots, Y_n$. In this case, $p = 1$. The MLE of σ^2 based on the $Y_1, \ldots, Y_n$ is $\sum_{i=1}^n (Y_i - \overline{Y})^2/n$, which is a (downward) biased estimator of σ^2. One set of linear combinations having the orthogonality property $\boldsymbol{C}\boldsymbol{F} = \boldsymbol{0}$ is obtained as follows. Let $\overline{Y}$ be the mean of $Y_1, \ldots, Y_n$. The linear combinations $W_1 = Y_1 - \overline{Y}, \ldots, W_{n-1} = Y_{n-1} - \overline{Y}$ each have mean zero and correspond to multiplying $\boldsymbol{Y}^n$ by an easily described $(n-1) \times n$ matrix $\boldsymbol{C}$ having full row rank $n - 1$. Maximizing the likelihood based on $W_1, \ldots, W_{n-1}$ and expressing the result in terms of $Y_1, \ldots, Y_n$ give

$$\widehat{\sigma^2} = \sum_{i=1}^{n} (Y_i - \overline{Y})^2/(n-1)\,. \tag{3.3.7}$$

The $n-1$ divisor in the error sum of squares (3.3.7) produces an unbiased estimator of σ^2. ♦

Returning to the general case (3.3.1), it can be shown that the REML estimator of $\boldsymbol{\kappa}$ is independent of the choice of linear combinations used to construct $\boldsymbol{W}$ subject to the number of columns of $\boldsymbol{C}$ being maximal in the sense of $\boldsymbol{C}$ having rank $n - p$ (Harville (1974), Harville (1977)). With some algebra it can be shown that the REML estimator of σ_Z^2 is

$$\widetilde{\sigma_Z^2} = \frac{n_s}{n_s - p}\widehat{\sigma_Z^2} = \frac{1}{n_s - p}\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right)^\top \boldsymbol{R}_{tr}^{-1}(\boldsymbol{\kappa})\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right), \tag{3.3.8}$$

where $\widehat{\sigma_Z^2}$ is the MLE of σ_Z^2 (see formula (3.3.5)) and the REML estimator of $\boldsymbol{\kappa}$ is the minimizer of

$$(n_s - p)\,\ell n\left(\widetilde{\sigma_Z^2}\right) + \ell n\,(\det(\boldsymbol{R}_{tr}(\boldsymbol{\kappa}))) + \ell n\left(\det\left(\boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1}(\boldsymbol{\kappa})\boldsymbol{F}_{tr}\right)\right). \tag{3.3.9}$$

The predictor of $y(\boldsymbol{x}^{te})$ corresponding to the REML estimator of $\widehat{\boldsymbol{\kappa}}$ is denoted as the REML-EBLUP of $y(\boldsymbol{x}^{te})$.

3.3.4 Cross-Validation EBLUPs

Cross-validation is a popular method for choosing model parameters in parametric model settings. Important early references describing cross-validation are Allen (1974), Stone (1974), and Stone (1977); Hastie et al. (2001) summarize recent applications.

Again assume that the correlation function is parametric with $R(\cdot) = R(\cdot\,|\,\boldsymbol{\kappa})$ so that $\boldsymbol{R}_{tr} = \boldsymbol{R}_{tr}(\boldsymbol{\kappa})$ and $\boldsymbol{r}_{te} = \boldsymbol{r}_{te}(\boldsymbol{\kappa})$. For $i = 1, \ldots, n_s$, let $\widehat{y}_{-i}(\boldsymbol{\kappa})$ denote the predictor (3.3.2) of $y(\boldsymbol{x}_i^{tr})$ when $\boldsymbol{\kappa}$ is the true correlation parameter based on all the data *except* $(\boldsymbol{x}_i^{tr}, y(\boldsymbol{x}_i^{tr}))$. The cross-validated estimator of $\boldsymbol{\kappa}$ minimizes the empirical mean squared prediction error:

$$\text{XV-PE}(\boldsymbol{\kappa}) \equiv \sum_{i=1}^{n_s} \left(\widehat{y}_{-i}(\boldsymbol{\kappa}) - y(\boldsymbol{x}_i^{tr})\right)^2.$$

More general forms of the cross-validation criterion have been proposed by Golub et al. (1979) and Wahba (1980). The predictor of $y(\boldsymbol{x}^{te})$ corresponding to the XV estimator of $\widehat{\boldsymbol{\kappa}}$ is denoted as the XV-EBLUP of $y(\boldsymbol{x}^{te})$.

3.3.5 Posterior Mode EBLUPs

The motivation and form of the posterior mode EBLUP are as follows. Recall from Theorem 3.1 that the minimum MSPE predictor of $Y(\boldsymbol{x}^{te})$ is $E[Y(\boldsymbol{x}^{te}) \,|\, \boldsymbol{Y}^{tr}]$. As described in Sect. 4.3, in fully Bayesian settings where a prior is available for $(\boldsymbol{\beta}, \sigma_Z^2, \boldsymbol{\kappa})$, this conditional mean is often calculated from

$$E\left[Y(\boldsymbol{x}^{te}) \,\middle|\, \boldsymbol{Y}^{tr}\right] = E\left[E\left[Y(\boldsymbol{x}^{te}) \,\middle|\, \boldsymbol{Y}^{tr}, \boldsymbol{\kappa}\right] \,\middle|\, \boldsymbol{Y}^{tr}\right], \tag{3.3.10}$$

where the inner expectation on the right-hand side of (3.3.10) is regarded as a function of $\boldsymbol{\kappa}$ and the outer expectation is with respect to the (marginal) posterior distribution $[\boldsymbol{\kappa} \,|\, \boldsymbol{Y}^{tr}]$. Sections 4.1 and 4.2 give several examples of closed-form expressions for $E[Y(\boldsymbol{x}^{te}) \,|\, \boldsymbol{Y}^{tr}, \boldsymbol{\kappa}]$. Even if $E[Y(\boldsymbol{x}^{te}) \,|\, \boldsymbol{Y}^{tr}, \boldsymbol{\kappa}]$ is known, the density of $[\boldsymbol{\kappa} \,|\, \boldsymbol{Y}^{tr}]$ generally cannot be expressed in closed form. One simple, but nevertheless attractive, approximation to $E[Y(\boldsymbol{x}^{te}) \,|\, \boldsymbol{Y}^{tr}]$ via the right-hand side of (3.3.10) is

$$E\left[Y(\boldsymbol{x}^{te}) \,\middle|\, \boldsymbol{Y}^{tr}, \widehat{\boldsymbol{\kappa}}\right], \tag{3.3.11}$$

where $\widehat{\boldsymbol{\kappa}}$ is the posterior mode of $[\boldsymbol{\kappa} \,|\, \boldsymbol{Y}^{tr}]$. This approximation is based on the (greatly) simplifying assumption that $[\boldsymbol{\kappa} \,|\, \boldsymbol{Y}^{tr}]$ is degenerate with mass located at its mode (Gibbs (1997)).

The line of reasoning in the previous paragraph suggests estimating $\boldsymbol{\kappa}$ by the $\widehat{\boldsymbol{\kappa}}$ that maximizes

$$\left[\boldsymbol{\kappa} \,\middle|\, \boldsymbol{Y}^{tr}\right] = \frac{[\boldsymbol{Y}^{tr} \,|\, \boldsymbol{\kappa}][\boldsymbol{\kappa}]}{[\boldsymbol{Y}^{tr}]} \propto \left[\boldsymbol{Y}^{tr} \,\middle|\, \boldsymbol{\kappa}\right][\boldsymbol{\kappa}]\,.$$

The *posterior mode* EBLUP of $y(\boldsymbol{x}^{te})$ is Eq. (3.3.11) where $\widehat{\boldsymbol{\kappa}}$ is the posterior mode of $\boldsymbol{\kappa}$; this EBLUP is denoted as the PMode-EBLUP.

While the predictor (3.3.11) uses the correlation parameter that seems "most likely" as judged by the posterior, choosing an informative prior for the $\boldsymbol{\kappa}$ parameters is often problematic. Indeed, the harried but Bayesian-inclined user may wish to compute the posterior mode $\widehat{\boldsymbol{\kappa}}$ based on a non-informative prior for $[\boldsymbol{\kappa}]$, as will be described in Sect. 4.3. The reader should be wary that the use of improper priors for correlation parameters can lead to improper posteriors. Berger et al. (2001) prove that, for isotropic correlation functions, many of the improper priors suggested in the literature yield improper posteriors. It also proposes an improper prior for default use whose posterior is proper.

3.3.6 Examples

Example 1.1 *(Continued)*. Recall the data introduced in Sect. 1.2 giving the computed time for a hot smoke layer (or smoke plume) to reach 5 ft above a fire source, which is denoted $y(\boldsymbol{x})$. The factors $\boldsymbol{x}$ which can affect $y(\boldsymbol{x})$ are the *height* and *area* of the room, its *heat loss fraction* (a measure of how well the room retains heat), and the *height of the fire source* above the room floor. The ranges for each input are listed in Table 3.1.

Input	Lower limit	Upper limit
Heat loss fraction	0.6	0.9
Height of the fire source	1.0	3.0
Room height	8.0	12.0
Room area	81.0	256.0

Table 3.1 Ranges of the factors in Example 1.1 that potentially affect the time for a smoke plume to reach 5 ft above a fire source

The same 40 points from a Sobol´ sequence that were considered earlier are now used as training data for prediction. Alternative choices of space-filling designs will be discussed in Chap. 5; indeed, the minimum interpoint distance of the $\binom{40}{2}$ pairs of 4-D points, when scaled to $[0, 1]^4$, is 0.22 versus the comparable value of 0.41 for a design that maximizes the minimum interpoint distance between all pairs of inputs (from a *maximin symmetric Latin hypercube design*, Sect. 5.3.3). Nevertheless Fig. 3.2 shows that the $6 = \binom{4}{2}$ two-dimensional projections of the 40 inputs for this design are visually spread.

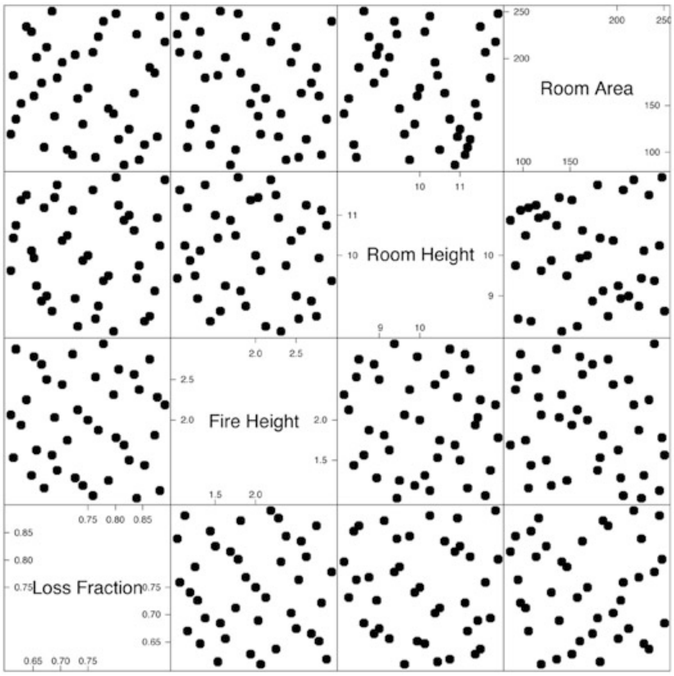


Fig. 3.2 Scatterplot matrix of the 40 input points used in Example 1.1

Figure 3.3 shows the marginal scatterplots of each of the four input variables versus the time for a fire to reach 5 ft above the fire source. Only *room area* appears to have a strong marginal relationship with response time.

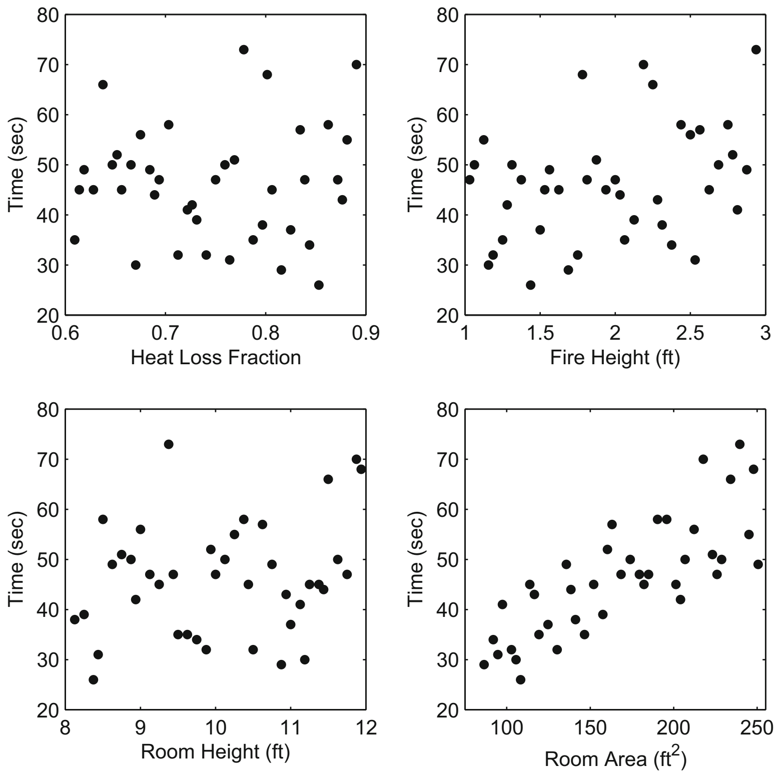


Fig. 3.3 Scatterplots of the time for a fire to reach 5 ft above a fire source versus each of the inputs: (1) heat loss fraction, (2) height of the fire source above the floor, (3) room height, and (4) room area, using the data from Example 1.1

Suppose it is desired to predict $y(\boldsymbol{x})$ for an $\boldsymbol{x}$ grid consisting of $320 = 4\times4\times4\times5$ equally spaced points over the ranges of the variables, `heat loss fraction`, `room height`, `fire height`, and `room area`, respectively. Consider REML-EBLUP prediction based on the stationary GP with constant mean β_0, process variance σ_Z^2, and Gaussian correlation function:

$$Cor[Y(\boldsymbol{x}_1), Y(\boldsymbol{x}_2)] = \exp\left\{-\sum_{j=1}^{d} \xi_j\,(x_{1,j} - x_{2,j})^2\right\}. \tag{3.3.12}$$

The results of a REML-EBLUP fit to these data are $\widehat{\beta_0} = 49.5666$, $\widehat{\sigma_Z^2} = 397.29$, and ξ_j estimates given in Table 3.2. Thus the rough "middle" of the time to reach

5 ft above the floor is 50 s, and the range of $y(\boldsymbol{x})$ is about 40 s ($40 \approx 2 \times \sqrt{397}$). Because the ranges of the inputs are quite different, it is problematic to interpret the estimated ξ_j. One method of providing an interpretation of the ξ_j is to place each input $(x_{1,j}, \ldots, x_{n_s,j})$ on the range [0, 1] by the transformation

$$x^s_{i,j} = \frac{x_{i,j} - \min_{1 \le i \le n_s} x_{i,j}}{r_j}$$

where $r_j = \max_{1 \le i \le n_s} x_{i,j} - \min_{1 \le i \le n_s} x_{i,j}$ is the range of the the j^{th} input. Then

$$Cor[Y(\boldsymbol{x}_1), Y(\boldsymbol{x}_2)] = \exp\left\{ - \sum_{j=1}^{d} \xi_j \, (x_{1,j} - x_{2,j})^2 \right\} = \exp\left\{ - \sum_{j=1}^{d} \xi_j \, r_j^2 \, (x^s_{1,j} - x^s_{2,j})^2 \right\}.$$

Thus $\exp\{-\xi_j \, r_j^2\}$ is the correlation between the times for two input configurations $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ that differ *only* in their j^{th} component and for which the j^{th} component values are at their extreme, i.e., $|x_{1,j} - x_{2,j}| = r_j$. Recalling that low correlations are associated with greater functional "activity," the fourth column of Table 3.2 shows that `room area` has far and away the greatest impact on $y(\boldsymbol{x})$ (is the most active input), that `room height` and `height of the fire source` have similar but much less activity than `room area`, and that `heat loss fraction` has very little impact on $y(\boldsymbol{x})$.

Input	$\widehat{\xi}_j$	$\widehat{\xi}_j r_j^2$	$\exp\left(-\widehat{\xi}_j \, r_j^2\right)$
Heat loss fraction	0.9852	0.0779	0.9250
Height of the fire source	0.0589	0.2141	0.8073
Room height	0.0122	0.1768	0.8380
Room area	0.0001	1.9343	0.1445

Table 3.2 REML estimates of the ξ_j, $j = 1, \ldots, 4$ in (3.3.12) and rough sensitivity measures, $\exp(-\xi_j \, r_j^2)$, for each input (smaller values associated with greater activity)

Numerous criteria can be used to measure the quality of the predictions at the 320 grid points. In this case, the true simulator output $y(\boldsymbol{x})$ was computed for the 320-point test grid. The visual assessment between the true and predicted times for the smoke plume to reach 5 ft above the ground is plotted in Fig. 3.4; there is a good agreement across the entire range of $y(\boldsymbol{x})$ values.

Cross-validation is a quantitative technique that is used when the prediction output is not available as would ordinarily be the case. Cross-validation is based on training data because the output is known for each training data input. The cross-validated prediction of $y(\boldsymbol{x}^{tr}_i)$ is obtained by omitting $(\boldsymbol{x}^{tr}_i, y(\boldsymbol{x}^{tr}_i))$ from the training data, fitting the model parameters using the remaining 39 points, and predicting $y(\boldsymbol{x}^{tr}_i)$ based on this fit. One can plot the cross-validated predictions versus the true values among other diagnostic methods. Here the cross-validated root mean squared prediction error of the 40-point $y(\boldsymbol{x})$ training data is calculated to be

$$\text{Cross-Val RMSPE } = 0.6012\,.$$

In this case the root mean squared prediction error can also be calculated for the 320-point test grid as

$$\sqrt{\frac{1}{320}\sum_{i=1}^{320}\left(y(\boldsymbol{x}_i^{te}) - \widehat{y}(\boldsymbol{x}_i^{te})\right)^2} = 0.4217\,.$$

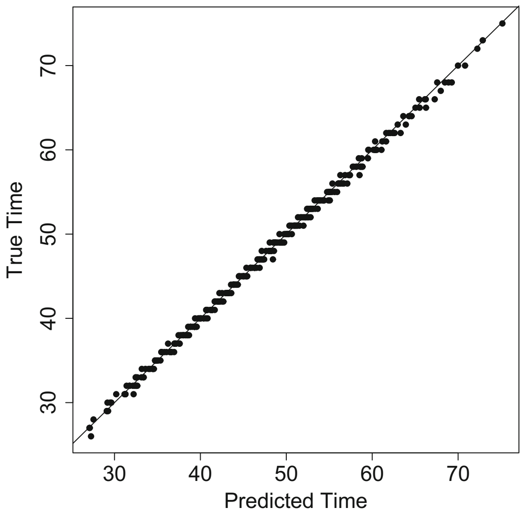


Fig. 3.4 Scatterplot of the true versus predicted times to reach 5 ft above a fire source for the equispaced grid of 320 points used in Example 1.1

The cross-validated root mean squared prediction error provides a very adequate, even worse-case, assessment of the predictive ability for this model and training data. ♦

3.4 A Simulation Comparison of EBLUPs

3.4.1 Introduction

Which correlation function should be selected when choosing an EBLUP based on a GP, and what method should be used to estimate the correlation parameters of this function? How many runs should be made of the simulator, assuming it is expensive, and what design should be used to collect the training data?

These questions will be addressed in this section. The reader should be aware that the discussion below on the strengths and drawbacks of various designs requires the reader to be *broadly familiar* with space-filling and other experimental designs. However, a detailed description of designs for simulator experiments is deferred until Chap. 5 and 6. Hence the reader may wish to postpone a *comprehensive* reading of the present section until completing these chapters. However much of the material of this section is self-contained, and the reader can glean basic answers to the choice of EBLUP as summarized in Sect. 3.4.4.

The section recommendations are based on our synthesis of previous studies of predictive accuracy together with our additional simulation study which also considers the empirical coverage of EBLUP-based prediction intervals.

3.4.2 A Selective Review of Previous Studies

There have been a number of previous studies of the predictive performance of EBLUPs. The list below is not exhaustive but contains representative articles: Bursztyn and Steinberg (2006), Liefvendahl and Stocki (2006), Ben-Ari and Steinberg (2007), Johnson et al. (2008), Jones and Johnson (2009), Johnson et al. (2010), Loeppky et al. (2010), Johnson et al. (2011), Williams et al. (2011), Silvestrini et al. (2013), Bachoc (2013), Atamturktur et al. (2013), Atamturktur et al. (2015), and Leatherman et al. (2018). The focus in this subsection is on the aspects of these papers that describe their predictive accuracy for test functions that are either (1) draws from a specified GP or (2) obtained by stochastically perturbing the coefficients of one (or more) given "central" analytic function(s) or (3) that contain only a few active inputs so that the function is (nearly) constant in all other inputs.

In addition, these studies differ in one or more of the following factors:

1. The predictors compared: most studies include EBLUPs as the featured predictor, but some also contrast standard parametric regression or nonparametric regression (such as multivariate adaptive regression splines);
2. For EBLUPs, differences in performance caused by:
   (a) the assumed correlation function of the GP model: usually one (or more) of the Gaussian, Matérn, and cubic correlations, and
   (b) the method of estimating the unknown correlation parameters: typically MLE, REML, or cross-validation;
3. The number of inputs, d;
4. The number of runs, n_s, and the experimental design used to form the training data: the design is typically one (or more) of:
   (a) maximin Latin hypercube designs (LHDs) and maximin designs (see Sects. 5.3 and 5.4), or
   (b) minimum average reciprocal distance (ARD) designs and minimum ARD LHDs (see Sects. 5.3 and 5.4), or

(c) designs that minimize the integrated mean squared prediction error (IMSPE)

$$\int \frac{E\left[(\widehat{y}(\boldsymbol{x}) - Y(\boldsymbol{x}))^2\right]}{\sigma_Z^2}\, d\boldsymbol{x}, \tag{3.4.1}$$

where $Y(\boldsymbol{x})$ is a GP with constant mean, process variance σ_Z^2, and Gaussian correlation function $R(\boldsymbol{h} \mid \boldsymbol{\rho}) = \prod_{i=1}^{d} \rho_i^{h_i^2}$ and $\widehat{y}(\boldsymbol{x})$ is the EBLUP given in (3.2.15) (of course, other correlation functions can be used; see Sect. 6.2.2), or

(d) uniform designs (Sect. 5.6.2);

5. The test bed of functions used to both construct the training data and measure the EBLUPs' predictive performance, either analytic functions selected to be "typical" of applications in a certain discipline or as noted in the introduction above;

6. The statistical performance measures used to judge predictor performance, usually either:

(a) the empirical root MSPE (ERMSPE)

$$\sqrt{\frac{1}{n_e}\sum_{i=1}^{n_e}\left(y(\boldsymbol{x}_i^{te}) - \widehat{y}(\boldsymbol{x}_i^{te})\right)^2} \tag{3.4.2}$$

obtained from a set of n_e evaluations of the test function $y(\boldsymbol{x})$ at inputs $\boldsymbol{x}_1^{te}, \ldots, \boldsymbol{x}_{n_e}^{te}$ and where $\widehat{y}(\boldsymbol{x})$ is a given EBLUP or other predictor based on training data from a particular design, or

(b) the empirical coverage of prediction intervals having a given nominal level, e.g., 95% nominal prediction limits

$$\text{ECov} = \frac{1}{n_e}\sum_{i=1}^{n_e} I\left\{y(\boldsymbol{x}_i^{te}) \in \widehat{y}(\boldsymbol{x}_i^{te}) \pm 1.96 \times \widetilde{\sigma_Z}\right\} \tag{3.4.3}$$

where $I\{E\}$ is 1 or 0 as the event E occurs or not,

$$\widetilde{\sigma_Z^2} = \frac{1}{n_s - 1}\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right)^\top \boldsymbol{R}_{tr}^{-1}(\widehat{\boldsymbol{\kappa}})\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right),$$

and $\widehat{\boldsymbol{\kappa}}$ is the estimated correlation parameter (vector) of the selected correlation function.

Some of the conclusions that have been reported in the literature are as follows:

1. Ben-Ari and Steinberg (2007) concluded that EBLUPs have superior prediction abilities compared with nonparametric regressors.

2. Liefvendahl and Stocki (2006) compared the predictive accuracy of maximin LHDs and minimum ARD LHDs in two test examples. They find the minimum ARD LHDs to be superior.

3. In a study of five design criteria used to predict draws from a stochastic process, Bursztyn and Steinberg (2006) found that two-level fractional factorial designs had consistently poor performance, while LHDs, lattice designs, and rotation designs had essentially equivalent empirical IMSPEs (see Beattie and Lin (1997), Bursztyn and Steinberg (2002), and the references therein). Other criteria led to different designs being selected as superior.

4. In a comparison of five experimental designs to predict functions drawn from a GP test bed with equally active inputs, Johnson et al. (2008) found that maximin LHDs and minimum IMSPE designs provided the smallest empirical average predicted variances.

5. When comparing designs using GP and cubic polynomial predictors Jones and Johnson (2009) found that GP predictors based on a maximin LHD predicted most accurately the values of the non-central F quantile test function with arguments: the quantile of interest, the two degrees of freedom, and the non-centrality parameter; the quantile function is the inverse of the non-central F cumulative distribution function. Jones and Johnson (2009) concluded that the GP which used training data determined by a maximin LHD had superior prediction accuracy compared to a cubic polynomial which used training data determined either by a maximin LHD or D-optimal design.

6. In a study of two-stage designs which they call predictive maturity, Loeppky et al. (2010) proposed to augment an initial design using batches of inputs selected from several different methods including two distance-based metrics and a process-based strategy. They compare their proposals to several intuitive schemes by predicting the outputs at grid of inputs for (1) the ($d = 2$) Branin function (see Example 6.2) and (2) draws from GPs with $d = 3$ and having inputs of unequal activity. They consider two performance criteria: the ERMSPE and the maximum prediction error over the prediction grid. They show that using either a maximin distance criterion, a weighted maximin distance criterion, or an entropy criterion (Sect. 6.2) provided the best improvement of the criteria considered. In related studies, Atamturktur et al. (2013) and Atamturktur et al. (2015) study the predictive maturity provided by batch augmented designs for validation experiments used to verify the predictive accuracy of simulators that have been developed for nuclear fuel design and other applications.

7. Johnson et al. (2011) compared the accuracy of EBLUP having either cubic or Gaussian correlation functions; they based the training data on specific space-filling designs and multiple run sizes. They measured the prediction accuracy of four test functions having $d \in \{2, 3, 8\}$ at a set of new, unevaluated input locations. Johnson et al. (2011) concluded that prediction based on the GP model with the Gaussian correlation function provides better accuracy than the cubic correlation

function, especially for larger sample sizes. None of the designs compared (maximin LHD, uniform, IMSPE optimal, maximin) was substantially more accurate than any other.

8. Bachoc (2013) compared the prediction accuracy of EBLUPs that use ML and cross-validation (XV) estimation of the correlation parameters for the Gaussian, the Matérn, and power exponential correlation families. See Sects. 3.3.2 and 3.3.4 for descriptions of ML and XV estimation, respectively, and Sects. 2.5 and 2.4 for definitions of the Matérn and power exponential correlation functions. The Bachoc (2013) comparisons are both theoretical and empirical, the latter using two test functions. Their studies conclude that XV estimation is more accurate than ML estimation; also the Gaussian correlation function produces smaller errors than does the power exponential function, while, for their examples, the Matérn family parameters had large estimated ν, and its predictions are similar to the Gaussian correlation function. As a small exception to this basic conclusion, the authors show that when predicting test surfaces which are draws from a GP with Gaussian correlation function, the ML-based EBLUP using the Gaussian correlation function has better prediction accuracy than the XV-based EBLUP using the Gaussian correlation function.

9. Leatherman et al. (2018) considered the *design* of a simulator experiment where input–output data had previously not been collected. Given a desired number of training data runs to be made, they compared two groups of space-filling LHDs (maximin LHDs, denoted by MmLHDs, and minimum ARD LHDs, denoted by mARDLHDs) and designs that minimize the local IMSPE-based criterion (3.4.1) for a given $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_d)$, among other designs. The training data from each design was used to determine an EBLUP based on a GP with Gaussian correlation function using REML to estimate the correlation parameters. Their comparison criterion was the ERMSPE in Eq. (3.4.2) computed using a space-filling set of inputs for each test function, $y(\boldsymbol{x})$. Among their conclusions were that, for small to moderate d ($d \in \{5, 8, 10\}$), sparse n_s ($n_s/d \in \{5, 10\}$), and smooth surfaces drawn from a single stationary process or a mixture of stationary processes, space-filling designs had worse prediction accuracy than locally IMSPE-optimal designs constructed for "small" $\boldsymbol{\rho}$, specifically $0.25 = \rho_1 = \cdots = \rho_d$ (denote this design by $L_{0.25}$). For larger d both maximin LHDs and $L_{0.25}$ performed equivalently and better than other designs. However, for nonstationary surfaces, depending on the location of the nontypical behavior, MmLHD and mARDLHD designs can yield smaller prediction errors than the locally optimal $L_{0.25}$ design or vice versa.

3.4.3 A Complementary Simulation Study of Prediction Accuracy and Prediction Interval Accuracy

This subsection adds our own simulation study of EBLUP performance to those described above. It uses two criteria to judge the performance of selected EBLUPs:

the empirical prediction accuracy and the empirical coverage of EBLUP-based prediction intervals.

As usual, the simulator output is denoted $y(x_1, \ldots, x_d)$. The EBLUPs considered below differed according to the following factors:

- the $Y(\boldsymbol{x})$ correlation function and method of estimating the correlation parameter:
  (a) Gaussian correlation function with $\boldsymbol{\kappa}$ estimated by REML (denoted by R-EBLUP-Gau in Figs. 3.7, 3.8, 3.9, 3.10, 3.11, and 3.12),
  (b) Gaussian correlation function with $\boldsymbol{\kappa}$ estimated by XV (denoted by X-EBLUP-Gau in Figs. 3.7, 3.8, 3.9, 3.10, 3.11, and 3.12),
  (c) Cubic correlation function with $\boldsymbol{\kappa}$ estimated by REML (denoted by R-EBLUP-Cub in Figs. 3.7, 3.8, 3.9, 3.10, 3.11, and 3.12),
  (d) Bohman correlation function with $\boldsymbol{\kappa}$ estimated by REML (denoted by R-EBLUP-Boh in Figs. 3.7, 3.8, 3.9, 3.10, 3.11, and 3.12).

  As is known from the variance components literature, REML estimation would presumably be superior to MLE if the fitted GP contained multiple regression parameters and hence MLE is not considered in this study.
- the number of runs per input: letting d denote the number of simulator inputs, $n_s = 10 \times d$ which is the recommendation of Loeppky et al. (2009)) *or* $n_s = 5 \times d$ which is a more difficult prediction setting.

Ultimately time considerations limited the number of combinations of EBLUP and test bed families that were considered. One limitation was that only maximin Latin hypercube designs were used in this study. The number of values that comprise each design is $n_s \times d$ which is large for $d = 20$ ($n_s \times d = 4000$ when $(n_s, d) = (200, 20)$). In such cases, any algorithm used to find maximin Latin hypercube designs will clearly produce an approximate design. Here the R package `slhd` was used to determine the designs used in this study.

3.4.3.1 Performance Measures

The performance measures (3.4.2) and (3.4.3) were used to compare the predictors run at a test set of n_e inputs. When $d = 2$ the $n_e = 625 = 25 \times 25$ equally spaced gridding of the input space was used, and when $d = 20$, a set of $n_e = 2000$ points which approximately maximized the minimum interpoint distance among the $\binom{2000}{2}$ pairs of rows (see Sect. 5.4) was selected.

3.4.3.2 Function Test Beds

Four test beds of functions were simulated, two of which contained $y(\boldsymbol{x})$ that appeared visually to be stationary and two of which contained $y(\boldsymbol{x})$ that were clearly nonstationary. The accuracy and coverage results were similar for the two visually stationary families and for the two nonstationary families. *Hence only one visually stationary and one nonstationary family will be presented here.*

In addition, d was taken to be either 2 (producing "easy" test bed problems), or $d = 20$ (yielding "hard" test bed problems). The (n_s, d) pairs used in this study are

$$(n_s, d) \in \{(10, 2), (20, 2), (100, 20), (200, 20)\}.$$

Test Bed #1: The first group of functions were "near-cubic" surfaces. When $d = 2$, the central surface was

$$\begin{aligned} y(x_1, x_2) = {} & x_1^3/3 - (R_1 + S_1)\, x_1^2/2 + (R_1 S_1)\, x_1 \\ & + x_2^3/3 - (R_2 + S_2)\, x_2^2/2 + (R_2 S_2) x_2 + A \sin\left(\frac{2\pi x_1 x_2}{Z}\right), \end{aligned} \tag{3.4.4}$$

where the six model coefficients (R_1, S_1), (R_2, S_2), and (A, Z) were mutually independent random variables. The values R_1 and S_1 were uniformly distributed over the interval $(0, 0.5)$ (denoted R_1 and $S_1 \sim U(0, 0.5)$), R_2 and $S_2 \sim U(0.5, 1.0)$, $A \sim U(0, 0.05)$, and $Z \sim U(0.25, 1.0)$. The additive sine term has a scale factor Z that provides between one and four oscillations in the product $x_1 x_2$ with an amplitude that ranges up to 0.05. The small amplitude coefficient of the $\sin(\cdot)$ term, A, assured that there were only minor deviations from the cubic model. Four $y(x_1, x_2)$ functions drawn using this stochastic mechanism are displayed in Fig. 3.5. These surfaces are smooth but can contain a significant interaction.

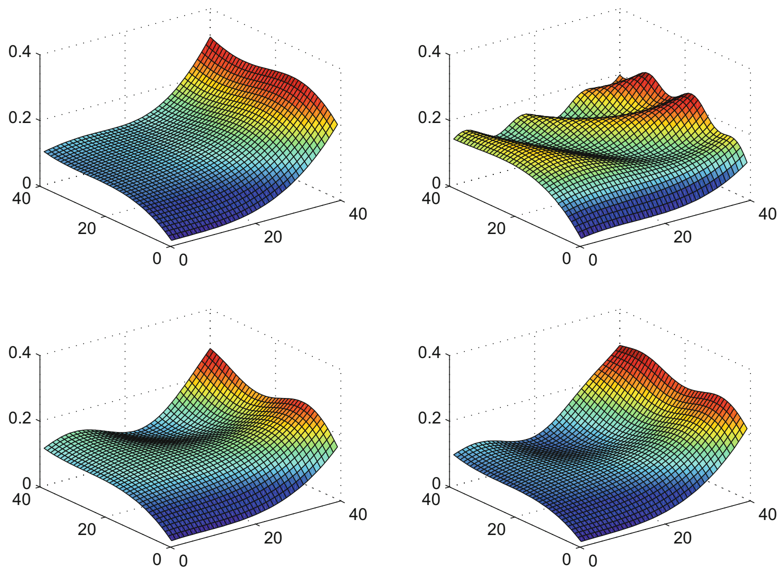


Fig. 3.5 Four of the 100 random $y(x_1, x_2)$ near-cubic surfaces (3.4.4)

When $d = 20$, the near-cubic test bed was generalized by adding sine terms for ten randomly selected pairs (x_i, x_j) with $i, j \in \{1, \ldots, 20\}$ and $i \neq j$ from the $\binom{20}{2}$

possible interaction pairs. This process yielded

$$y(\boldsymbol{x}) = \sum_{i=1,\ i\ odd}^{20} \left\{ x_i^3/3 - (R_{1i} + S_{1i})\, x_i^2 + (R_i\, S_i)\, x_i \right\}$$

$$+ \sum_{i=1,\ i\,even}^{20} \left\{ x_i^3/3 - (R_{2i} + S_{2i})\, x_i^2 + (R_{2i}\, S_{2i})\, x_i \right\}$$

$$+ \sum_{j=1}^{10} \left\{ A_j * \sin\left(\frac{2\pi \times x_{\pi(1,j)} x_{\pi(2,j)}}{Z} \right) \right\}, \qquad (3.4.5)$$

where all stochastic terms are mutually independent, each R_{1i} and S_{1i} is $U(0, 0.5)$, $i \in \{1, 3, \ldots, 19\}$, each R_{2i} and S_{2i} is $U(0.5, 1.0)$, $i \in \{2, 4, \ldots, 20\}$, $A_j \sim U(0, 0.05)$, and $Z_j \sim U(0.25, 1)$, $1 \leq j \leq 10$. The pairs $(\pi(1, j), \pi(2, j))$, $1 \leq j \leq 10$, are randomly selected without replication from the $\binom{20}{2}$ distinct subsets of size 2 from $\{1, \ldots, 20\}$.

Test Bed #2: The second group of test bed functions is a scaled and centered version of the functions considered by Ba and Joseph (2012) and earlier by Xiong et al. (2007); they are referred to as XB functions below. In brief, XB functions are smooth but have different behavior in the middle of the input range than near its edges. Hence they represent a significant challenge for stationary interpolation models. The members of this test bed have the form

$$y(\boldsymbol{x}) = C \prod_{i=1}^{d} \left\{ \sin\left(A_i (z_i - B_i)^4 \right) \times \cos\left(2z_i - B_i\right) + ((z_i - B_i)/2) \right\}, \qquad (3.4.6)$$

with $z_i = |x_i - 0.5|$ and $\boldsymbol{x} \in [0, 1]^d$. A stochastic test bed was formed from (3.4.6) by taking $\{A_i\}_{i=1}^d$ to be i.i.d $U(20, 35)$ and independent of $\{B_i\}_{i=1}^d$ which are i.i.d. $U(0.5, 0.9)$. The constant C was selected so that the vast majority of function draws varied over ± 3. Here C was set equal to 10 for $d = 2$, and C was set equal to 10^{13} for $d = 20$. To better understand the members of this test bed, Fig. 3.6 plots four XB $y(x_1, x_2)$ functions drawn using this stochastic mechanism. The center of the functions behaved differently than the edges, and thus $y(x_1, x_2)$ is more challenging to predict than function draws shown in Fig. 3.5.

3.4.3.3 Prediction Simulations

Figures 3.7, 3.8, 3.9, 3.10, 3.11, and 3.12 plot empirical RMSPEs and coverages over the set of simulation factors. Good EBLUPs have low ERMSPEs and ECovs close to their nominal 95% level. The large and small d cases provide different

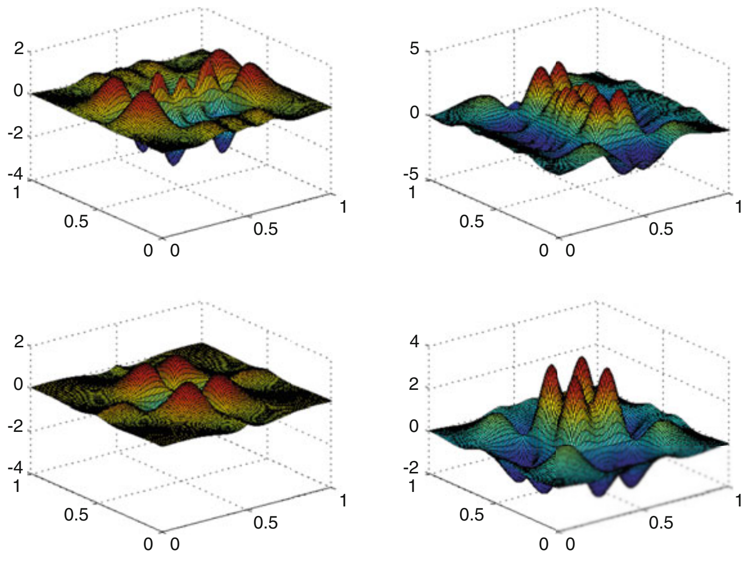


Fig. 3.6 Four of the 100 random $y(x_1, x_2)$ XB functions

perspectives on the EBLUPs studied as do the stationary and more difficult nonstationary surfaces.

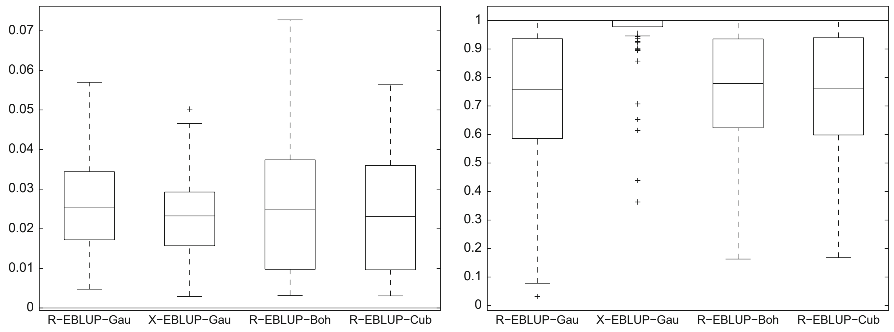


Fig. 3.7 For $(n_s, d) = (10, 2)$, left panel shows the distribution of ERMSPE for four EBLUPs over 100 near-cubic test functions (3.4.4) where each ERMSPE is based on 625 inputs in $[0, 1]^2$; right panel shows the distribution of 100 ECov values for the four EBLUPs where each ECov is the proportion of 625 inputs in $[0, 1]^2$ contained in nominal 95% prediction intervals

While each EBLUP method has strengths, the performance over all test beds must be synthesized to provide recommended EBLUPs. For example, looking at Fig. 3.8 alone suggests that for predictions from stationary "looking" surfaces with $(d, n) = (20, 2)$, the REML based on either the Gaussian, cubic, or Bohman correlations has slightly better ERMSPE than the cross-validated Gaussian correlation

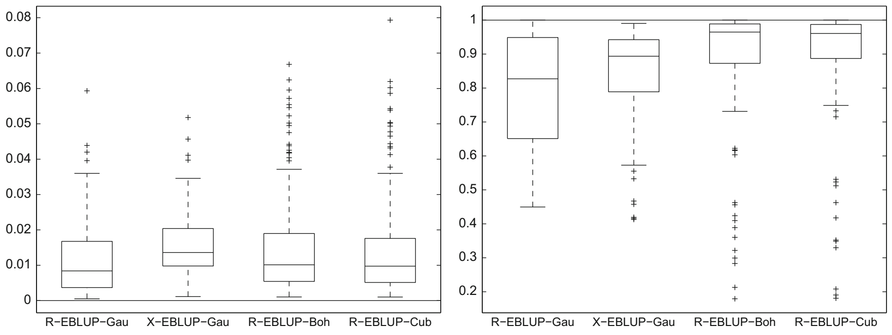


Fig. 3.8 For $(n_s, d) = (20, 2)$, left panel shows the distribution of ERMSPE for four EBLUPs over 100 near-cubic test functions (3.4.4) where each ERMSPE is based on 625 inputs in $[0, 1]^2$; right panel shows the distribution of 100 ECov values for the four EBLUPs where each ECov is the proportion of 625 inputs in $[0, 1]^2$ contained in nominal 95% prediction intervals

function. However the empirical coverages, with a few exceptions, are better for the compactly supported correlation functions. Views across the set of figures suggest that REML based on cubic and Bohman correlation functions are comparable for virtually all cases studied. For some of the larger d cases, only the cubic correlation function was considered. After study of the entire set of performance measures, we make the following recommendations.

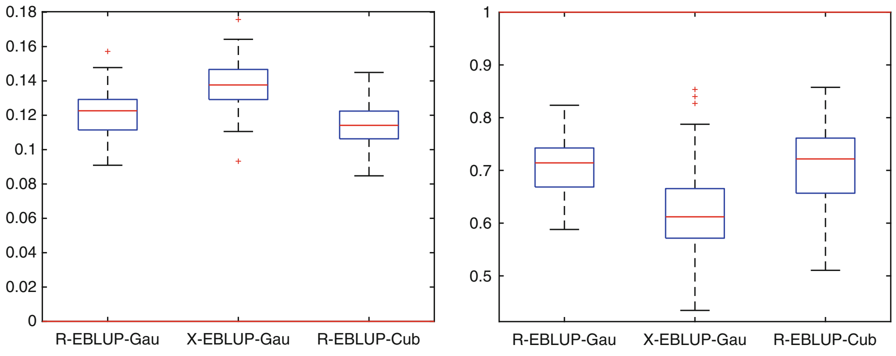


Fig. 3.9 For $(n_s, d) = (100, 20)$, left panel shows the distribution of ERMSPE for three EBLUPs over 100 near-cubic test functions (3.4.5) where each ERMSPE is based on 2000 inputs in $[0, 1]^{20}$; right panel shows the distribution of 100 ECov values for the three EBLUPs where each ECov is the proportion of the 2000 inputs in $[0, 1]^{20}$ contained in nominal 95% prediction intervals

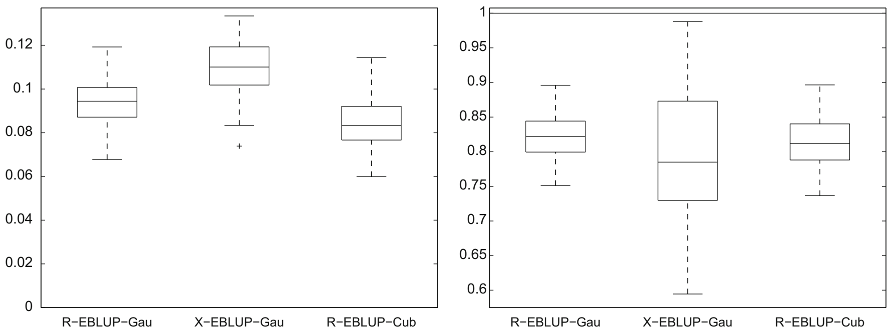


Fig. 3.10 For $(n_s, d) = (200, 20)$, Left Panel shows the distribution of ERMSPE for three EBLUPs over 100 near-cubic test functions (3.4.5) where each ERMSPE is based on 2000 inputs in $[0, 1]^{20}$; Right Panel shows the distribution of 100 ECov values for the three EBLUPs where each ECov is the proportion of the 2000 inputs in $[0, 1]^{20}$ contained in nominal 95% prediction intervals

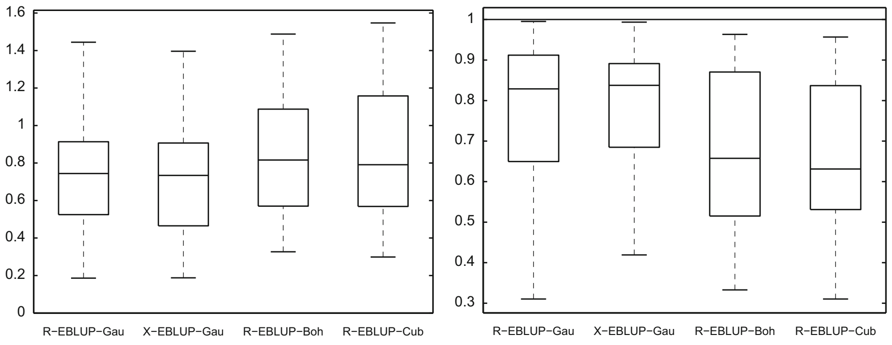


Fig. 3.11 For $(n_s, d) = (20, 2)$, left panel shows the distribution of ERMSPE for four EBLUPs over 100 XB test functions (3.4.6) where each ERMSPE is based on 625 inputs in $[0, 1]^2$; right panel shows the distribution of the 100 ECov values for the four EBLUPs where each ECov is the proportion of the 625 inputs in $[0, 1]^2$ contained in nominal 95% prediction intervals

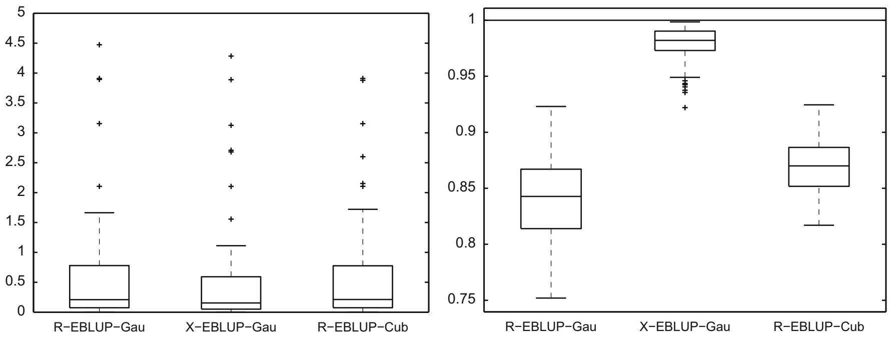


Fig. 3.12 For $(n_s, d) = (200, 20)$, left panel shows the distribution of ERMSPE for three EBLUPs over 100 XB test functions (3.4.6) where each ERMSPE is based on 2000 inputs in $[0, 1]^{20}$; right panel shows the distribution of 100 ECov values where each ECov is the proportion of the 2000 inputs in $[0, 1]^{20}$ contained in nominal 95% prediction intervals

3.4.4 Recommendations

1. For predicting smooth surfaces (including ones that can be viewed as draws from a stationary process) having a "small" number of inputs d (say, $d \leq 5$) based on $n \leq 10d$ training runs, the *XV-EBLUP with Gaussian correlation function* is recommended because this EBLUP produces prediction intervals for $y(\boldsymbol{x})$ whose empirical coverages are closer to their nominal coverages than other EBLUPs, while its ERMSPE prediction accuracy is comparable to those of other EBLUPs.
2. For predicting smooth surfaces having a "large" number of inputs d (say, $d \geq 15$) based on $n \geq 10d$ training runs, the *REML-EBLUP with Gaussian correlation function* is recommended because this EBLUP produces prediction intervals for $y(\boldsymbol{x})$ whose empirical coverages are closer to their nominal coverages and are more tightly clustered about their empirical mean coverage than those of other EBLUPs, while the ERMSPE of this EBLUP is slightly smaller than those of other EBLUPs.
3. For predicting "complicated" functions, such as XB surfaces, having a "small" number of inputs d (say, $d \leq 5$), either the *XV-EBLUP with Gaussian correlation function* or the *REML-EBLUP with Gaussian correlation function* is recommended. When the function to be predicted has a "large" number of inputs d (say, $d \geq 15$), then the *XV-EBLUP with Gaussian correlation function* is recommended. As above, these recommendations are based on the empirical closeness of the EBLUP-based prediction intervals for $y(\boldsymbol{x})$ to their nominal values, the tightness of the empirical coverages about their mean, and the smallness of the ERMSPEs.

3.5 EBLUP Prediction of Multivariate Simulator Output

This section derives EBLUPs when multiple outputs are produced from each set of inputs. One setting where this occurs is when several simulators are available for computing the *same* response as, for example, when there are both "fast" (less accurate) and "slow" (more accurate) codes to compute an output. Such a hierarchy of codes is natural when, for example, finite element models of varying mesh sizes can be used to implement a mathematical model (Kennedy and O'Hagan (2000); Qian and Wu (2008)).

Another application where multiple outputs occur is in Pareto optimization. In such an application, the different outputs represent trade-off characteristics of a "system" defined by $\boldsymbol{x}$, e.g., the lift, strength, and weight of an aircraft wing. Pareto optimization is applied when it is desired to minimize simultaneously a set of outputs $y_1(\boldsymbol{x}), \ldots, y_m(\boldsymbol{x})$. Typically there is no $\boldsymbol{x}^{opt}$ that simultaneously minimizes all outputs. The goal of Pareto optimization is to find the set of all inputs $\boldsymbol{x}$ for which all inputs are not *dominated* simultaneously. Technically this means that the objective is to identify all $\boldsymbol{x}$ for which there does *not exist* another input, say $\boldsymbol{x}^\star$ for which $y_i(\boldsymbol{x}^\star) \le y_i(\boldsymbol{x})$ for all i, with strict inequality for some i.

A third setting that produces multiple outputs is when both $y(\cdot)$ and its partial derivatives ("adjoints") are available. Many engineering codes are of this type. The derivatives provide information about the $y(\cdot)$ surface. Example 3.7 considers this situation.

Using the models introduced in Sect. 2.5, Sect. 3.5.1 uses these models to describe predictors for one of the several computed responses. Detailed examples are given to conclude the section in 3.5.2. These multiple response models will be used again in Sect. 6.3.4, where an algorithm will be presented that locates a minimizing $\boldsymbol{x}_{\min}$ of $y_1(\boldsymbol{x})$ that satisfies feasibility constraints defined by $y_2(\cdot), \ldots, y_m(\cdot)$.

3.5.1 Optimal Predictors for Multiple Outputs

For the sake of definiteness, consider the problem of predicting $y_1(\cdot)$ at the input $\boldsymbol{x}^{te}$, given output from all m codes, where each code has been evaluated at its own unique set of training inputs (possibly the same set). To describe the training data, for each output i, $1 \le i \le m$, let $\boldsymbol{x}^i_\ell$, $1 \le \ell \le n_i$, denote the set of inputs at which $y_i(\cdot)$ has been computed, and let $\boldsymbol{y}^{n_i}_i = (y_i(\boldsymbol{x}^i_1), \ldots, y_i(\boldsymbol{x}^i_{n_i}))^\top$ denote the $n_i \times 1$ vector of all evaluations of $y_i(\cdot)$.

From the results of Sect. 3.2, the best MSPE predictor of $y_1(\boldsymbol{x}^{te})$ based on this training data is

$$\widehat{y}_1(\boldsymbol{x}^{te}) = E\left[Y^{te}_1 \,\middle|\, \boldsymbol{Y}^{n_1}_1 = \boldsymbol{y}^{n_1}_1, \ldots, \boldsymbol{Y}^{n_m}_m = \boldsymbol{y}^{n_m}_m\right], \tag{3.5.1}$$

where $Y^{te}_1 = Y_1(\boldsymbol{x}^{te})$ and $\boldsymbol{Y}^{n_i}_i = (Y_i(\boldsymbol{x}^i_1), \ldots, Y_i(\boldsymbol{x}^i_{n_i}))^\top$, for $1 \le i \le m$. It is primarily a bookkeeping problem to set up the proper identifications in the notation of these earlier sections. *The explicit formula for the conditional expectation (3.5.1) depends on the joint distribution of* $(Y_1(\boldsymbol{x}^{te}), \boldsymbol{Y}^{n_1}_1, \ldots, \boldsymbol{Y}^{n_m}_m)$. The simplest cases for which one can derive a formula for (3.5.1) are Gaussian models. First consider the (one-stage) Gaussian model (3.5.2).

The one-stage Gaussian model assumes that all mean and covariance parameters are known; it specifies the joint distribution of $(Y_0, \boldsymbol{Y}^{n_1}_1, \ldots, \boldsymbol{Y}^{n_m}_m)$ to be the multivariate normal distribution:

$$\begin{pmatrix} Y_0 \\ \boldsymbol{Y}^{n_1}_1 \\ \vdots \\ \boldsymbol{Y}^{n_m}_m \end{pmatrix} \sim N_{1+\sum_{i=1}^m n_i}\left(\boldsymbol{F}\boldsymbol{\beta}\,,\, \sigma^2_1 \boldsymbol{R}^\star\right), \tag{3.5.2}$$

where $\boldsymbol{F}$ and $\boldsymbol{R}^\star$ are defined by

$$\begin{bmatrix} \boldsymbol{f}_1^\top(\boldsymbol{x}_0) & \cdots & \boldsymbol{0}_{1\times p_m} \\ \boldsymbol{F}_1 & \cdots & \boldsymbol{0}_{n_1\times p_m} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0}_{n_m\times p_1} & \cdots & \boldsymbol{F}_m \end{bmatrix} \text{ and}$$

$$\begin{bmatrix} 1 & \boldsymbol{r}_1^\top & \tau_2\,\boldsymbol{r}_{12}^\top & \cdots & \tau_m\,\boldsymbol{r}_{1m}^\top \\ \boldsymbol{r}_1 & \boldsymbol{R}_1 & \tau_2\,\boldsymbol{R}_{12} & \cdots & \tau_m\,\boldsymbol{R}_{1m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \tau_m\,\boldsymbol{r}_{1m} & \tau_m\,\boldsymbol{R}_{1m}^\top & \tau_2\,\tau_m\,\boldsymbol{R}_{2m}^\top & \cdots & \tau_m^2\,\boldsymbol{R}_m \end{bmatrix},$$

respectively, where

- $\tau_i = \sigma_i/\sigma_1$, $2 \le i \le m$,
- $\boldsymbol{f}_1(\boldsymbol{x}_0)$ is the $p_1 \times 1$ vector of regressors for $Y_1(\cdot)$ at $\boldsymbol{x}_0$,
- $\boldsymbol{F}_i = \left(\boldsymbol{f}_i^\top(\boldsymbol{x}_\ell^i)\right)$ is the $n_i \times p_i$ matrix of regressors for the n_i inputs, where $y_i(\cdot)$ is evaluated for $1 \le i \le m$, and $1 \le \ell \le n_i$,
- $\boldsymbol{\beta} = (\boldsymbol{\beta}_1^\top, \ldots, \boldsymbol{\beta}_m^\top)^\top$, where $\boldsymbol{\beta}_i$ is the $p_i \times 1$ vector of regression coefficients associated with $\boldsymbol{Y}_i^{n_i}$, $1 \le i \le m$,
- $\boldsymbol{R}_i$ is the $n_i \times n_i$ matrix of correlations among the elements of $\boldsymbol{Y}_i^{n_i}$, $1 \le i \le m$,
- $\boldsymbol{r}_1 = \left(R_1(\boldsymbol{x}_0 - \boldsymbol{x}_1^1), \ldots, R_1(\boldsymbol{x}_0 - \boldsymbol{x}_{n_1}^1)\right)^\top$ is the $n_1 \times 1$ vector of correlations of $Y_1(\boldsymbol{x}_0)$ with $\boldsymbol{Y}_1^{n_1}$,
- $\boldsymbol{r}_{1i} = \left(R_{1i}(\boldsymbol{x}_0 - \boldsymbol{x}_1^i), \ldots, R_{1i}(\boldsymbol{x}_0 - \boldsymbol{x}_{n_i}^i)\right)^\top$ is the $n_i \times 1$ vector of correlations of $Y_1(\boldsymbol{x}_0)$ with $\boldsymbol{Y}_i^{n_i}$, $2 \le i \le m$,
- $\boldsymbol{R}_{ij}$ is the $n_i \times n_j$ matrix of correlations between $\boldsymbol{Y}_i^{n_i}$ and $\boldsymbol{Y}_j^{n_j}$, $1 \le i < j \le m$.

By Sect. 3.2, the conditional expectation (3.5.1) is given by

$$\boldsymbol{f}_0^\top\boldsymbol{\beta} + \boldsymbol{r}_0^\top\boldsymbol{R}^{-1}\,(\boldsymbol{y}^n - \boldsymbol{F}\boldsymbol{\beta}) \tag{3.5.3}$$

with the identifications $\boldsymbol{f}_0^\top = \left[\boldsymbol{f}_1^\top(\boldsymbol{x}_0)\ \ \boldsymbol{0}_{1\times(p-p_1)}\right]$, $\boldsymbol{r}_0^\top = \left[\boldsymbol{r}_1^\top\ \ \tau_2\boldsymbol{r}_{12}^\top\ \ \cdots\ \ \tau_m\boldsymbol{r}_{1m}^\top\right]$, $\boldsymbol{R}$ is the bottom right $(\sum_{i=1}^m n_i)\times(\sum_{i=1}^m n_i)$ submatrix of $\boldsymbol{R}^\star$, $\boldsymbol{y}^n$ is the $(\sum_{i=1}^m n_i)\times 1$ vector of observed outputs, and $\boldsymbol{F}$ and $\boldsymbol{\beta}$ are as in (3.5.2). In practice, the predictor (3.5.3) can seldom be employed because it requires knowledge of marginal correlation functions, the joint correlation functions, *and* the ratio of all the process variances.

However, *empirical versions* of (3.5.3) *are* of practical use. As in Sect. 3.3.2, assume that each of the correlation matrices $\boldsymbol{R}_i$, $1 \le i \le m$, and cross-correlation matrices $\boldsymbol{R}_{ij}$, $1 \le i < j \le m$, is known up to a finite vector of parameters. Suppose that $\boldsymbol{\kappa}_i$ is the vector of unknown parameters for $R_i(\cdot)$ and $\boldsymbol{\kappa}_{ij}$ is the unknown parameter vector for $R_{ij}(\cdot)$. Then $\boldsymbol{\kappa} = (\tau_2, \ldots, \tau_m, \boldsymbol{\kappa}_1, \ldots, \boldsymbol{\kappa}_m, \boldsymbol{\kappa}_{12}, \ldots, \boldsymbol{\kappa}_{m-1,m})$ contains all the unknown parameters required to describe the correlations of $Y_1(\boldsymbol{x}_0)$ with the training data.

As sketched in Sect. 4.3, a fully Bayesian predictor for this setup puts a prior on $[\boldsymbol{\beta}, \sigma_1^2, \boldsymbol{\kappa}]$ and uses the mean of the predictive distribution (3.5.1) as the desired predictor. For multiple response models, it is even more difficult analytically and numerically to construct this predictor than in the single response case, and we have previously noted that the single response case can be very difficult indeed (Handcock and Stein (1993)).

As in Sect. 3.3.2, we use the predictor

$$\widehat{y}_1(\boldsymbol{x}_0) = E\left[Y_0 \,\middle|\, \boldsymbol{Y}_1^{n_1} = \boldsymbol{y}_1^{n_1}, \ldots, \boldsymbol{Y}_m^{n_m} = \boldsymbol{y}_m^{n_m}, \widehat{\boldsymbol{\kappa}}\right],$$

where $\widehat{\boldsymbol{\kappa}}$ is estimated from $(\boldsymbol{Y}_1^{n_1}, \ldots, \boldsymbol{Y}_m^{n_m})$ based on the Gaussian likelihood (or restricted likelihood) induced from (3.5.2).

As developed in Sect. 3.4, this predictor can be obtained by first considering a two-stage model in which $\boldsymbol{\kappa}$ is *known*. Suppose that the conditional distribution of $(Y_0, \boldsymbol{Y}_1^{n_1}, \ldots, \boldsymbol{Y}_m^{n_m})$ given $(\boldsymbol{\beta}, \sigma_1^2)$ is (3.5.2) and the marginal distribution of $(\boldsymbol{\beta}, \sigma_1^2)$ is the (non-informative) prior:

$$[\boldsymbol{\beta}, \sigma_1^2] \propto \frac{1}{\sigma_1^2}.$$

The predictor, $E[Y_0 \,|\, \boldsymbol{Y}_1^{n_1} = \boldsymbol{y}_1^{n_1}, \ldots, \boldsymbol{Y}_m^{n_m} = \boldsymbol{y}_m^{n_m}, \boldsymbol{\kappa}]$, corresponding to this two-stage model is

$$\widehat{y}_1(\boldsymbol{x}_0) = \boldsymbol{f}_0^\top \widehat{\boldsymbol{\beta}} + \boldsymbol{r}_0^\top \boldsymbol{R}^{-1}(\boldsymbol{Y}^n - \boldsymbol{F}\widehat{\boldsymbol{\beta}}), \tag{3.5.4}$$

where $\boldsymbol{f}_0$, $\boldsymbol{r}_0$, $\boldsymbol{R}$, $\boldsymbol{Y}^n$, and $\boldsymbol{F}$ are described following (3.5.3) and $\widehat{\boldsymbol{\beta}}$ is the generalized least squares estimator of $\boldsymbol{\beta}$ based on $\boldsymbol{Y}^n$. When $\boldsymbol{\kappa}$ is *unknown*, we estimate $\boldsymbol{\kappa}$ in (3.5.4) using MLE or REML to produce an EBLUP of $Y_1(\boldsymbol{x}_0)$.

A final approach to modeling output is implicit from the models with univariate output from simulators with both quantitative and qualitative inputs which are discussed in Sect. 2.4. The basic model for output with qualitative and quantitative inputs is given in (2.4.1) and can be viewed as a special case of the model for multivariate output described in Sect. 2.5. Comparing the models in (2.4.1) and (2.5.1), one observes that they look similar with $Y(\boldsymbol{x}, i)$ and $f_j(\boldsymbol{x}, i)$ in (2.4.1) being special cases of $Y_i(\boldsymbol{x})$ and $f_{i,j}(\boldsymbol{x})$ in (2.5.1). However, univariate output from a simulator with qualitative and quantitative variables has features that differ from multivariate simulator output. In models with quantitative and qualitative inputs, the output $Y(\boldsymbol{x}, i)$, for each value of i, represents the same measured property (e.g., failure depth in the context of Example 1.2). For general multivariate output, the $Y_i(\boldsymbol{x})$ may represent very different measured quantities with possibly different units and scales. For general multivariate output, for each $\boldsymbol{x}$, there are m responses $Y_1(\boldsymbol{x}), \ldots, Y_m(\boldsymbol{x})$. For univariate output with quantitative and qualitative inputs, $Y(\boldsymbol{x}, i)$ need not be observed at the same values of $\boldsymbol{x}$ as $Y(\boldsymbol{x}, j)$. Univariate output with quantitative and qualitative variables looks more like general multivariate output if for each value of the qualitative variable, we observe the response at the same set of inputs $\boldsymbol{x}$.

3.5.2 Examples

Example 3.6 (A Simple Analytic Example). Consider predicting

$$y_g(x) = e^{-1.4x}\cos(7\pi x/2), \quad x \in [0,1]$$

based on a set of evaluations of $y_g(\cdot)$ and

$$y_p(x) = y_g(x/(2-x)), \quad x \in [0,1]\,.$$

The pair of functions is plotted in Fig. 3.13. A total of 7 training points were selected for $y_g(x)$ and 11 training points for $y_p(x)$; five x input sites were common to the $y_g(x)$ and $y_p(x)$ training data sets. Figure 3.13 denotes the locations of the $y_p(x)$ inputs by open triangles and the locations of the $y_g(x)$ inputs by filled triangles. By plotting the inputs along the bottom horizontal axis, the figure highlights the relationship between the two input sets. The input set for $y_g(x)$ does not include any $x \in (0, 0.1)$ corresponding to its steeply decreasing, left-most section. In contrast, the input $x = 0.05$ for $y_p(\boldsymbol{x})$ is in the region of steep $y_g(x)$ decrease. The accuracy of predicting $y_g(\cdot)$ for an EBLUP based on the $y_g(\cdot)$ data *alone* is compared with the accuracy of an EBLUP that uses *both* the $y_g(x)$ and $y_p(x)$ training data.

There are many possible bivariate models that can be proposed for describing $y_g(x)$ and $y_p(x)$. This example illustrates prediction based on the constructive model of Kennedy and O'Hagan (2000), who considered predicting the outcome of a finite element code that uses a very fine grid based on output from the fine grid code and from the output of a faster but less accurate code that is based on a coarser grid. In this example, $y_p(\cdot)$ represents the output from the fast, but *poorer*, code and $y_g(\cdot)$ the output from the slow, but *good*, code. *Our goal is to predict the output of the good code at 100 equispaced points over (0,1).*

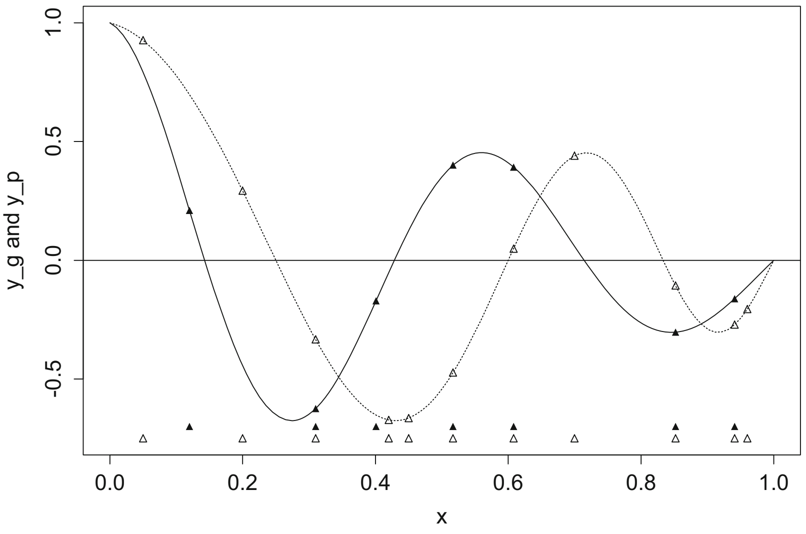


Fig. 3.13 The functions $y_g(x)$ (solid) and $y_p(x)$ (dashed) on [0, 1]. The open triangles denote locations of training data for $y_p(x)$, and filled triangles denote locations of training data for $y_g(x)$

Consider the GP model for this setting in which $y_p(\boldsymbol{x})$ is regarded as a draw from

$$Y_p(\boldsymbol{x}) = \boldsymbol{f}_p^\top(\boldsymbol{x})\boldsymbol{\beta}_p + Z_p(\boldsymbol{x}), \tag{3.5.5}$$

where the regression function $\boldsymbol{f}_p^\top(\boldsymbol{x})$ specifies the large-scale, nonstationary structure of $y_p(\boldsymbol{x})$ and $Z_p(\cdot)$ is a stationary Gaussian process that determines the local features of the code; $Z_p(\boldsymbol{x})$ is assumed to have zero mean, variance σ_p^2, and correlation function $R_p(\cdot)$.

The more accurate code, $y_g(\boldsymbol{x})$, is regarded as a draw from

$$Y_g(\boldsymbol{x}) = \boldsymbol{f}_g^\top(\boldsymbol{x})\boldsymbol{\beta}_g + \rho\, Z_p(\boldsymbol{x}) + Z_a(\boldsymbol{x}). \tag{3.5.6}$$

The relationship between the slow and fast codes can be made in at least two ways. First, if the same regressors are used for both $y_p(\boldsymbol{x})$ and $y_g(\boldsymbol{x})$, i.e., $\boldsymbol{f}_p^\top(\boldsymbol{x}) = \boldsymbol{f}_g^\top(\boldsymbol{x})$, then similar large-scale variation will be present in both $y_g(\boldsymbol{x})$ and $y_p(\boldsymbol{x})$. Second, the small-scale variation of $y_p(\boldsymbol{x})$, $Z_p(\boldsymbol{x})$, is also present in $Y_g(\boldsymbol{x})$, perhaps with a scale adjustment. Finally, $Z_a(\boldsymbol{x})$ represents an "accuracy enhancement" in the good code. Assume that the process $Z_a(\boldsymbol{x})$ is a stationary Gaussian process with variance σ_a^2 and correlation function $R_a(\cdot)$, independent of $Z_p(\boldsymbol{x})$.

The variances of $Y_p(\boldsymbol{x})$ and $Y_g(\boldsymbol{x})$ and their covariance are computed as

$$\begin{aligned}
Cov\left[Y_p(\boldsymbol{x}_1), Y_p(\boldsymbol{x}_2)\right] &= Cov\left[Z_p(\boldsymbol{x}_1), Z_p(\boldsymbol{x}_2)\right] \\
&= \sigma_p^2\, R_p\,(\boldsymbol{x}_1 - \boldsymbol{x}_2) \\
Cov\left[Y_g(\boldsymbol{x}_1), Y_p(\boldsymbol{x}_2)\right] &= Cov\left[\rho\, Z_p(\boldsymbol{x}_1) + Z_a(\boldsymbol{x}_1), Z_p(\boldsymbol{x}_2)\right] \\
&= \sigma_p^2\, \rho\, R_p\,(\boldsymbol{x}_1 - \boldsymbol{x}_2) \\
Cov\left[Y_g(\boldsymbol{x}_1), Y_g(\boldsymbol{x}_2)\right] &= Cov\left[\rho\, Z_p(\boldsymbol{x}_1) + Z_a(\boldsymbol{x}_1), \rho\, Z_p(\boldsymbol{x}_2) + Z_a(\boldsymbol{x}_2)\right] \\
&= \rho^2 Cov\left[Z_p(\boldsymbol{x}_1), Z_p(\boldsymbol{x}_2)\right] \;+\; Cov\left[Z_a(\boldsymbol{x}_1), Z_a(\boldsymbol{x}_2)\right] \\
&= \sigma_p^2\left(\rho^2\, R_p\,(\boldsymbol{x}_1 - \boldsymbol{x}_2) + \tau^2\, R_a\,(\boldsymbol{x}_1 - \boldsymbol{x}_2)\right),
\end{aligned}$$

where $\tau^2 = \sigma_a^2/\sigma_p^2$. Thus the variance of $Y_p(\boldsymbol{x})$ is $Cov[Y_p(\boldsymbol{x}), Y_p(\boldsymbol{x})] = \sigma_p^2$, and the variance of $Y_g(\boldsymbol{x})$ is $Cov[Y_g(\boldsymbol{x}), Y_g(\boldsymbol{x})] = \sigma_p^2\,(\rho^2 + \tau^2)$. The correlation functions of $Y_g(\cdot)$ and $Y_p(\cdot)$ are

$$R_g(\boldsymbol{h}) \equiv \frac{\rho^2\, R_p(\boldsymbol{h}) + \tau^2\, R_a(\boldsymbol{h})}{\rho^2 + \tau^2} \;\text{ and }\; R_p(\boldsymbol{h}),$$

respectively, while their cross-correlation function is

$$R_{12}(\boldsymbol{h}) \equiv \frac{\rho\, R_p(\boldsymbol{h})}{\sqrt{\rho^2 + \tau^2}}.$$

The components of the $y_g(\boldsymbol{x})$ predictor in (3.5.4) are now straightforward to identify.

To gain a feel for this model, consider $(Y_p(\boldsymbol{x}), Y_g(\boldsymbol{x}))$ draws from (3.5.5) to (3.5.6) when both $Y_p(\boldsymbol{x})$ and $Y_g(\boldsymbol{x})$ have mean zero, $\sigma_a^2 = 1.0 = \sigma_p^2$, and

$$R_a(w) = R_p(w) = \exp(-5.0 \times w^2)\,.$$

Figure 3.14 plots $(Y_p(\boldsymbol{x}), Y_g(\boldsymbol{x}))$ draws corresponding to four choices of ρ. Notice how the two functions "move together" for positive ρ and in opposite directions for negative ρ with the strength of the effect increasing in ρ. The draws in Fig. 3.14 suggest that, for the appropriate choice of parameters, this model may reasonably produce data of the form shown in Fig. 3.13.

The EBLUPs for the data represented in Fig. 3.13 are based on a *constant mean* regression model having the power exponential correlation function

$$R_\ell(h) = \exp\left(-\xi_\ell\, |h|^{p_\ell}\right)\,,$$

with $\xi_\ell > 0$ and $0 < p_\ell \leq 2$ for $\ell \in \{p, a\}$. The EBLUP below uses a REML correlation parameter estimate.

Table 3.3 lists the empirical root mean squared prediction error of the two $y_g(x)$ predictors at a 100-point equispaced grid on (0, 1). The $y_g(x)$ predictor that uses *both*

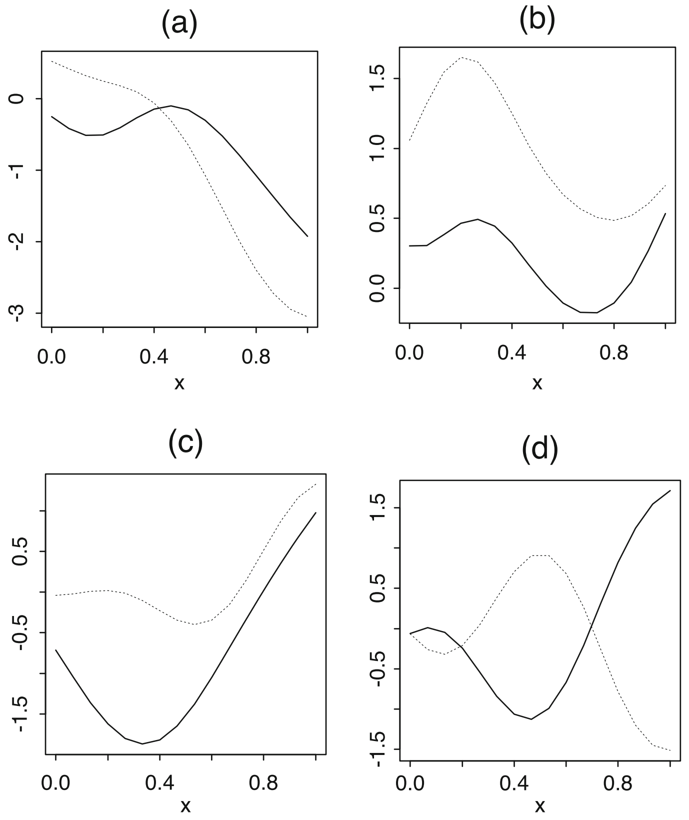


Fig. 3.14 Draws of $Y_p(\boldsymbol{x})$ (solid) and $Y_g(\boldsymbol{x})$ (dashed) from the bivariate Gaussian process (3.5.5)–(3.5.6); panel (**a**) uses $\rho = 0.2$, panel (**b**) uses $\rho = 0.8$, panel (**c**) uses $\rho = -0.2$, and panel (**d**) uses $\rho = -0.8$

outputs is a great improvement over that based on the $y_g(x)$ training data alone; the ERMSPE is improved fivefold by using the $y_p(x)$ information. Figure 3.15 plots the predicted $y_g(x)$ and the predictors based on the REML estimates of the correlation parameters. Clearly, the $y_p(x)$ training data near the origin allows us to indirectly "see" the sharp negative slope in $y_g(x)$ in this region. ♦

$y_g(\boldsymbol{x})$ data only	$y_g(\boldsymbol{x})$ & $y_p(\boldsymbol{x})$ data
0.155	0.031

Table 3.3 Empirical root mean squared prediction errors of two REML-EBLUP predictors of $y_g(\boldsymbol{x})$: $y_g(\boldsymbol{x})$ training data only versus combined $y_g(\boldsymbol{x})$ and $y_p(\boldsymbol{x})$ training data

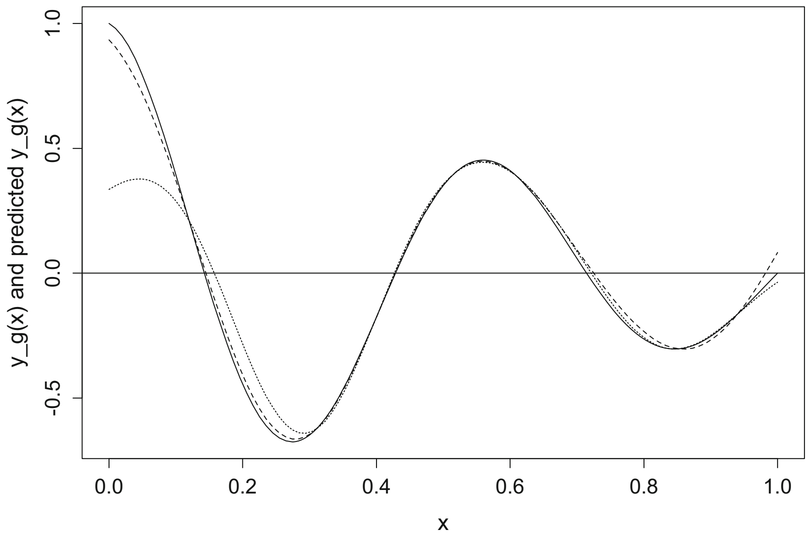


Fig. 3.15 The target function $y_g(x)$ (solid), the EBLUP based on $y_g(x)$ (dotted), and the EBLUP based on $(y_p(x), y_g(x))$ (dashed) with correlation parameters estimated by REML

Example 3.7 (Prediction with Derivative Information). Consider a code having bivariate input $\boldsymbol{x} = (x_1, x_2)$ that also produces the partial derivates of $y(\cdot)$,

$$y^{(1)}(\boldsymbol{x}) = \partial y(\boldsymbol{x})/\partial x_1 \quad \text{and} \quad y^{(2)}(\boldsymbol{x}) = \partial y(\boldsymbol{x})/\partial x_2 \,.$$

Assume the multivariate GP model for a function and its partial derivatives given in Sect. 2.5.3 with product power exponential correlation function

$$R(h_1, h_2) = \exp\{-\xi_1\, h_1^2\} \times \exp\{-\xi_2\, h_2^2\} \,.$$

Let $\boldsymbol{x}_1 = (x_{1,1}, x_{1,2})$ and $\boldsymbol{x}_2 = (x_{2,1}, x_{2,2})$. Applying the formulas (2.5.10) and (2.5.11) gives the pairwise joint covariances of $Y(\cdot)$, $Y^{(1)}(\cdot)$, and $Y^{(2)}(\cdot)$:

$$Cov\left[Y(\boldsymbol{x}_1), Y^{(j)}(\boldsymbol{x}_2)\right] = -2\,\xi_j\,(x_{1,j} - x_{2,j})\,\sigma_Z^2\, R(\boldsymbol{x}_1 - \boldsymbol{x}_2)\,,$$

$$Cov\left[Y^{(j)}(\boldsymbol{x}_1), Y^{(j)}(\boldsymbol{x}_2)\right] = \left(2\,\xi_j - 4\,\xi_j^2\,(x_{1,j} - x_{2,j})^2\right)\,\sigma_Z^2\,R(\boldsymbol{x}_1 - \boldsymbol{x}_2)$$

for $j = 1, 2$, and

$$Cov\left[Y^{(1)}(\boldsymbol{x}_1), Y^{(2)}(\boldsymbol{x}_2)\right] = 4\,\xi_1\,\xi_2\,(x_{1,1} - x_{2,1})\,(x_{1,2} - x_{2,2})\,\sigma_Z^2\,R(\boldsymbol{x}_1 - \boldsymbol{x}_2)\,.$$

Using these covariance functions, the EBLUP based on $y(\boldsymbol{x})$, $y^{(1)}(\boldsymbol{x})$, and $y^{(2)}(\boldsymbol{x})$ can be computed from (3.5.4) with appropriate code to estimate the scale parameters (ξ_1, ξ_2) and to implement the predictor.

As a specific numerical example, let

$$y(x_1, x_2) = 2x_1^3 x_2^2$$

on $[-1, 1]^2$, which is displayed in Fig. 3.16. The cubic and quadratic characters of

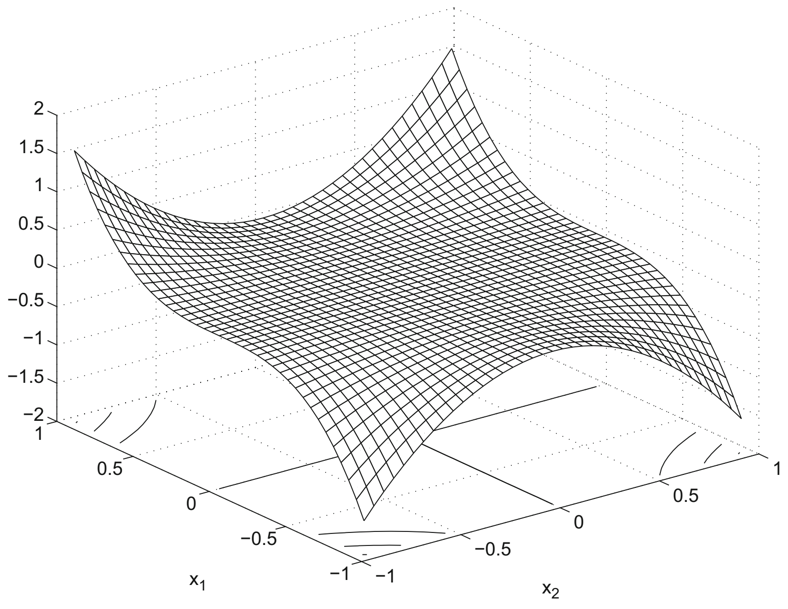


Fig. 3.16 The function $y(x_1, x_2) = 2x_1^3 x_2^2$ on $[-1, 1]^2$

$y(\cdot)$ in x_1 and x_2, respectively, are clearly visible on this domain. The first partial derivatives of $y(\cdot)$ are

$$y^{(1)}(x_1, x_2) = \partial y(x_1, x_2)/\partial x_1 = 6x_1^2 x_2^2$$

and

$$y^{(2)}(x_1, x_2) = \partial y(x_1, x_2)/\partial x_2 = 4x_1^3 x_2\,.$$

Consider the 14-point training data displayed in Fig. 3.17. This set of locations has the intuitive feature that it allows us to "learn" about $y(\boldsymbol{x})$ and its partial derivatives over the the entire input space—the design is "space-filling." Space-filling designs are discussed in detail in Chap. 5.

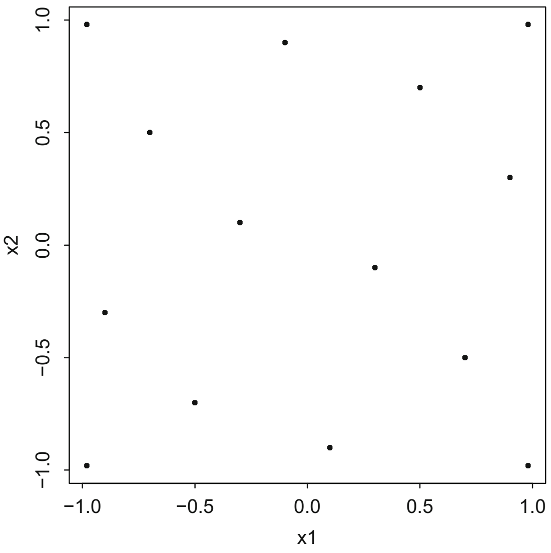


Fig. 3.17 Fourteen-point training design on $[-1, 1]^2$

We illustrate the benefit of adding the derivative information by predicting $y(\cdot)$ on the 1521 ($= 39^2$) grid of points $[-0.95\ (0.05)\ 0.95]^2$ by first using the predictor of $y(\cdot)$ based on $y(\cdot)$ *alone* and then by using the predictor of $y(\cdot)$ based on $(y(\cdot), y^{(1)}(\cdot), y^{(2)}(\cdot))$. The regression function for $Y(\cdot)$ is taken to be constant, β_0. We measure the accuracy of the generic predictor $\widehat{y}(\cdot)$ of $y(\cdot)$ by its *empirical root mean squared prediction error* (ERMSPE) at the 1521-point test grid, which is defined to be

$$\text{ERMSPE}(\widehat{y}) = \sqrt{\frac{1}{1521}\sum_{i=1}^{1521}\left(y(\boldsymbol{x}_i) - \widehat{y}(\boldsymbol{x}_i)\right)^2}.$$

Table 3.4 summarizes the estimated model parameters and ERMSPEs for the two predictors. Figure 3.18 displays the predictor of $y(\cdot)$ based on the 14-point training set evaluations of $y(\cdot)$; this predictor has ERMSPE equal to 0.3180. Figure 3.19 plots the predictor of $y(\cdot)$ based on the 14-point training set evaluations of $(y(\cdot), y^{(1)}(\cdot), y^{(2)}(\cdot))$; it has ERMSPE equal to 0.2566. The fit based on $y(\cdot)$ and its derivatives *both* is more visually appealing (compare prediction of the x_1 edges with those of the true surface in Fig. 3.16) *and* has a 24% smaller ERMSPE than that based on $y(\cdot)$ alone. ♦

	Predictor based on	
	$y(\cdot)$	$\left(y(\cdot), y^{(1)}(\cdot), y^{(2)}(\cdot)\right)$
$\widehat{\xi}_1$	0.7071	13.4264
$\widehat{\xi}_2$	10.9082	8.7060
$\widehat{\beta}_0$	0.00	0.00
ERMSPE	0.3180	0.2566

Table 3.4 Empirical RMSPEs at the 39^2 points on the grid $[-0.95\ (0.05)\ 0.95]^2$ for the EBLUPs based on $y(\cdot)$ alone and $(y(\cdot), y^{(1)}(\cdot), y^{(2)}(\cdot))$

Example 3.8 (An Analysis Based on a Qualitative and Quantitative (QQ) Input Model). This example, courtesy of the Procter & Gamble Company, illustrates the analysis of computer experiments with QQ inputs. Engineers at the Procter & Gamble Company conducted a computer experiment to model a particular part of an oral care packing line. The computer simulation involved nine continuous variables and one three-level qualitative variable representing three types of a particular equipment part. The engineers used a 132-run optimal sliced Latin hypercube design with 44 runs at each of the three levels of the categorical variable. See Sect. 5.3.2 for an introduction to sliced Latin hypercube designs and Ba et al. (2015) for optimal sliced Latin hypercube designs, including the example discussed here.

The model of Qian et al. (2008) and Zhou et al. (2011) for qualitative and quantitative inputs, discussed in Sect. 2.4, was assumed. In order to use standard software for fitting a stationary Gaussian process model, the method in Zhang (2014) was em-

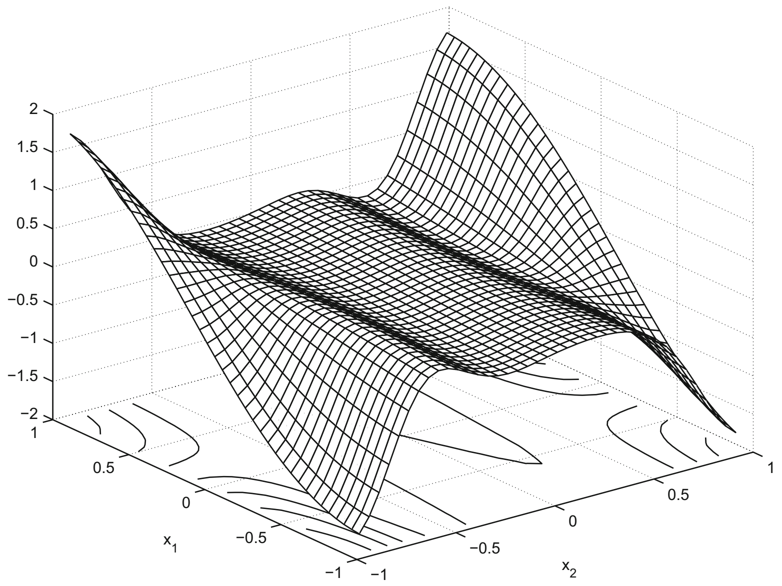


Fig. 3.18 Prediction surface based on $y(\cdot)$ using the 14-point exploratory design and the power exponential correlation function

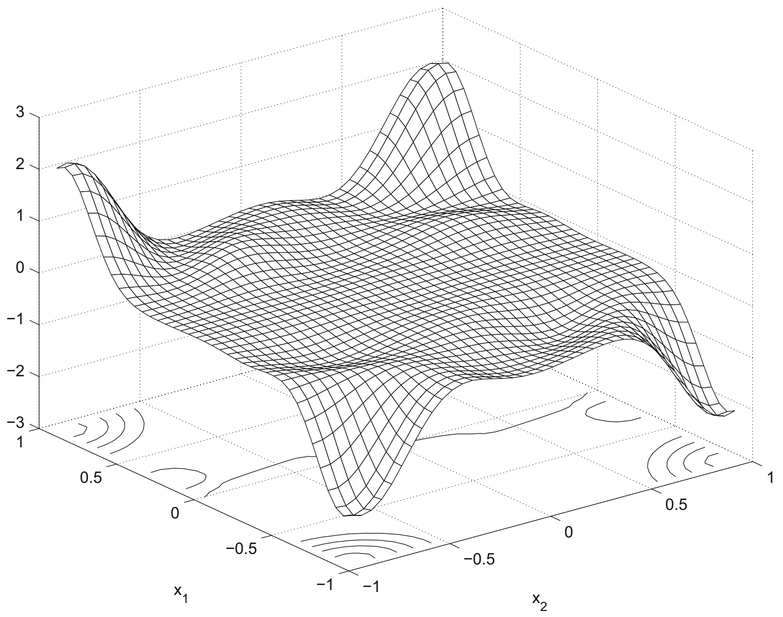


Fig. 3.19 Prediction surface based on $(y(\cdot), y^{(1)}(\cdot), y^{(2)}(\cdot))$ using the 14-point exploratory design and the exponential correlation function

ployed. There are $T = 3$ levels of the qualitative variable, and hence, as described in Sect. 2.4 (above (2.4.4)), one must create variables $W_{1,1}(i)$, $W_{1,2}(i)$, and $W_{2,2}(i)$, $1 \le i \le 3$, to represent the qualitative variable. Following the procedure described above (2.4.4), the resulting values for these variables as a function of i are:

$$W_{1,1}(i) : W_{1,1}(1) = 1\,,\ W_{1,1}(2) = W_{1,1}(3) = 0\,,$$
$$W_{1,2}(i) : W_{1,2}(1) = W_{1,2}(2) = 1\,,\ W_{1,2}(3) = 0\,,\ \text{and}$$
$$W_{2,2}(i) : W_{2,2}(1) = W_{2,2}(3) = 0\,,\ W_{2,2}(2) = 1\,.$$

These are treated as three additional *quantitative* variables. Assuming a constant mean function and a Gaussian correlation function, a stationary Gaussian process model (3.1.1), with 12 quantitative predictor variables (the original nine plus $W_{1,1}$, $W_{1,2}$, and $W_{2,2}$), was fit to the data using the software package JMP. In place of JMP, one could use any software that fits a stationary Gaussian process model with constant mean function and Gaussian correlation function. The estimates of the correlation parameters $\xi^*_{1,1}$, $\xi^*_{1,2}$, and $\xi^*_{2,2}$ corresponding to $W_{1,1}$, $W_{1,2}$, and $W_{2,2}$ were found to be $\xi^*_{1,1} = 0.0011942$, $\xi^*_{1,2} = 221.98592$, and $\xi^*_{2,2} = 0.117032$. Using the expressions given below (2.4.4), the estimates of the cross-correlation parameters $\tau_{i,j}$ in (2.4.3) are $\tau_{1,2} = 0.8885$, $\tau_{1,3} \approx 0$, and $\tau_{2,3} \approx 0$. The response surfaces corresponding to the qualitative variable at values 1 and 2 are strongly correlated, but neither is correlated with the response surface corresponding to the qualitative variable at value 3.

A plot of the JMP jackknife predictions versus the actual values of the response is shown in Fig. 3.20, with the points corresponding to the three values of the qualitative variable represented by different plotting symbols with unique colors. As can be seen in Fig. 3.20, the response surface corresponding to level 3 of the qualitative variable has the worst fit. The root mean squared prediction errors of the jackknife predictor are

All Curves	0.4233
For Curve 1	0.3162
For Curve 2	0.2885
For Curve 3	0.5953

.

These values show that the QQ model has similar predictive ability for Curves 1 and 2, while the outputs for Curve 3 are not predicted as well ♦

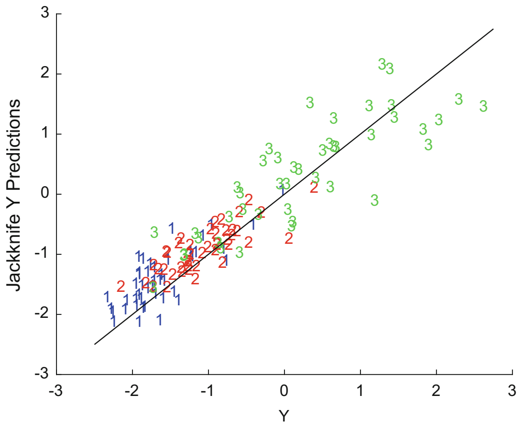


Fig. 3.20 Plot of the jackknife predicted versus actual responses. The points are labeled 1 (blue), 2 (red), and 3 (green) according to the value of the qualitative variable

3.6 Chapter Notes

3.6.1 Proof That (3.2.7) Is a BLUP

Assume that the $Y^{te} = Y(\boldsymbol{x}^{te})$ and the training data $\boldsymbol{Y}^{tr}$ satisfy

$$\begin{pmatrix} Y^{te} \\ \boldsymbol{Y}^{tr} \end{pmatrix} \sim N_{1+n_s}\left(\begin{pmatrix} \boldsymbol{f}_{te}^{\top} \\ \boldsymbol{F}_{tr} \end{pmatrix} \boldsymbol{\beta},\ \sigma_Z^2 \begin{bmatrix} 1 & \boldsymbol{r}_{te}^{\top} \\ \boldsymbol{r}_{te} & \boldsymbol{R}_{tr} \end{bmatrix} \right)$$

where the correlation matrices $\boldsymbol{R}_{tr}$ and $\boldsymbol{r}_{te}$ are known. Because the argument in (3.2.17) shows that unbiased requires that $a_0 + \boldsymbol{a}^\top \boldsymbol{F}_{tr}\boldsymbol{\beta} = \boldsymbol{f}_{te}^\top\boldsymbol{\beta}$ must hold for all $\boldsymbol{\beta}$, the linear predictor $\widehat{y}(\boldsymbol{x}^{te}) = a_0 + \boldsymbol{a}^\top \boldsymbol{Y}^{tr}$ is unbiased for $Y(\boldsymbol{x}^{te})$ provided

$$a_0 = 0 \quad \text{and} \quad \boldsymbol{F}_{tr}^\top \boldsymbol{a} = \boldsymbol{f}_{te}\,. \tag{3.6.1}$$

In particular, algebra shows that

$$\begin{aligned}\boldsymbol{b}^\top = \Big[&\boldsymbol{f}_{te}^\top(\boldsymbol{F}_{tr}^\top\boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr})^{-1}\boldsymbol{F}_{tr}^\top\boldsymbol{R}_{tr}^{-1}\\ &+ \boldsymbol{r}_{te}^\top\boldsymbol{R}_{tr}^{-1}\left(\boldsymbol{I}_{n_s} - \boldsymbol{F}_{tr}(\boldsymbol{F}_{tr}^\top\boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr})^{-1}\boldsymbol{F}_{tr}^\top\right)\boldsymbol{R}_{tr}^{-1}\Big]\end{aligned} \tag{3.6.2}$$

used to construct $\widehat{y}(\boldsymbol{x}^{te})$ in (3.2.7) satisfies the unbiased condition.

Now fix an arbitrary LUP of $Y(\boldsymbol{x}^{te})$, say $\boldsymbol{a}^\top\boldsymbol{Y}^{tr}$. Let $\boldsymbol{Z}^{tr} \equiv \boldsymbol{Y}^{tr} - \boldsymbol{F}_{tr}\boldsymbol{\beta}$ and $Z^{te} \equiv Y^{te} - \boldsymbol{f}_{te}^\top\boldsymbol{\beta}$ be the test and training GP values centered to have mean zero. Then the MSPE of $\boldsymbol{a}^\top\boldsymbol{Y}^{tr}$ is

$$\begin{aligned}E\left[\left(\boldsymbol{a}^\top\boldsymbol{Y}^{tr} - Y^{te}\right)^2\right] &= E\left[\left(\boldsymbol{a}^\top(\boldsymbol{F}_{tr}\boldsymbol{\beta} + \boldsymbol{Z}^{tr}) - (\boldsymbol{f}_{te}^\top\boldsymbol{\beta} + Z^{te})\right)^2\right]\\ &= E\left[\left((\boldsymbol{a}^\top\boldsymbol{F}_{tr} - \boldsymbol{f}_{te}^\top)\boldsymbol{\beta} + \boldsymbol{a}^\top\boldsymbol{Z}^{tr} - Z^{te}\right)^2\right]\\ &= E\left[\boldsymbol{a}^\top\boldsymbol{Z}^{tr}(\boldsymbol{Z}^{tr})^\top\boldsymbol{a} - 2\boldsymbol{a}^\top\boldsymbol{Z}^{tr}Z^{te} + (Z^{te})^2\right] \qquad (3.6.3)\\ &= \sigma_Z^2\boldsymbol{a}^\top\boldsymbol{R}_{tr}\boldsymbol{a} - 2\sigma_Z^2\boldsymbol{a}^\top\boldsymbol{r}_{te} + \sigma_Z^2\\ &= \sigma_Z^2\left(\boldsymbol{a}^\top\boldsymbol{R}_{tr}\boldsymbol{a} - 2\boldsymbol{a}^\top\boldsymbol{r}_{te} + 1\right), \qquad (3.6.4)\end{aligned}$$

where (3.6.3) follows from (3.6.1). Thus the BLUP chooses $\boldsymbol{a}$ to minimize

$$\boldsymbol{a}^\top\boldsymbol{R}_{tr}\boldsymbol{a} - 2\boldsymbol{a}^\top\boldsymbol{r}_{te} \tag{3.6.5}$$

subject to

$$\boldsymbol{F}_{tr}^\top\boldsymbol{a} = \boldsymbol{f}_{te}\,. \tag{3.6.6}$$

The method of Lagrange multipliers can be used to minimize the quadratic objective function (3.6.5) subject to linear constraints (3.6.6). We find $(\boldsymbol{a}, \boldsymbol{\lambda}) \in \mathbb{R}^{n+p}$ to minimize

$$\boldsymbol{a}^\top\boldsymbol{R}_{tr}\boldsymbol{a} - 2\boldsymbol{a}^\top\boldsymbol{r}_{te} + 2\boldsymbol{\lambda}^\top(\boldsymbol{F}_{tr}^\top\boldsymbol{a} - \boldsymbol{f}_{te})\,. \tag{3.6.7}$$

Calculating the gradient of (3.6.7) with respect to $(\boldsymbol{a}, \boldsymbol{\lambda})$ and setting it equal to the zero vector give the system of equations:

$$\begin{aligned}\boldsymbol{F}_{tr}^\top\boldsymbol{a} - \boldsymbol{f}_{te} &= \boldsymbol{0}\\ \boldsymbol{R}_{tr}\boldsymbol{a} - \boldsymbol{r}_{te} + \boldsymbol{F}_{tr}\boldsymbol{\lambda} &= \boldsymbol{0}\end{aligned}$$

or

$$\begin{bmatrix}\boldsymbol{0} & \boldsymbol{F}_{tr}^\top\\ \boldsymbol{F}_{tr} & \boldsymbol{R}_{tr}\end{bmatrix}\begin{pmatrix}\boldsymbol{\lambda}\\ \boldsymbol{a}\end{pmatrix} = \begin{pmatrix}\boldsymbol{f}_{te}\\ \boldsymbol{r}_{te}\end{pmatrix},$$

which implies, using formula (B.5.1) for the inverse of a block matrix, that

$$\begin{aligned}\begin{pmatrix}\boldsymbol{\lambda}\\ \boldsymbol{a}\end{pmatrix} &= \begin{bmatrix}\mathbf{0} & \boldsymbol{F}_{tr}^\top\\ \boldsymbol{F}_{tr} & \boldsymbol{R}_{tr}\end{bmatrix}^{-1}\begin{pmatrix}\boldsymbol{f}_{te}\\ \boldsymbol{r}_{te}\end{pmatrix}\\ &= \begin{bmatrix}-\boldsymbol{Q}^{-1} & \boldsymbol{Q}^{-1}\boldsymbol{F}_{tr}^\top\boldsymbol{R}_{tr}^{-1}\\ \boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr}\boldsymbol{Q}^{-1} & \boldsymbol{R}_{tr}^{-1} - \boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr}\boldsymbol{Q}^{-1}\boldsymbol{F}_{tr}^\top\boldsymbol{R}_{tr}^{-1}\end{bmatrix}\begin{pmatrix}\boldsymbol{f}_{te}\\ \boldsymbol{r}_{te}\end{pmatrix},\end{aligned}$$

where $\boldsymbol{Q} = \boldsymbol{F}_{tr}^\top\boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr}$. After a small amount of algebra, the $\boldsymbol{a}$ solution is seen to be (3.6.2). ■

3.6.2 Derivation of Formula 3.2.8

From the MSPE (3.6.4) and using (3.6.2), it must be shown that

$$\sigma_Z^2\left(1 + \boldsymbol{b}^\top\boldsymbol{R}_{tr}\boldsymbol{b} - 2\boldsymbol{b}^\top\boldsymbol{r}_{te}\right) = \sigma_Z^2\left(1 - \boldsymbol{r}_{te}^\top\boldsymbol{R}_{tr}^{-1}\boldsymbol{r}_{te} + \boldsymbol{h}^\top\boldsymbol{Q}^{-1}\boldsymbol{h}\right),$$

where $\boldsymbol{Q} = \boldsymbol{F}_{tr}^\top\boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr}$, $\boldsymbol{h} = \boldsymbol{f}_{te} - \boldsymbol{F}_{tr}^\top\boldsymbol{R}_{tr}^{-1}\boldsymbol{r}_{te}$, and

$$\boldsymbol{b}^\top = \boldsymbol{f}_{te}^\top\boldsymbol{Q}^{-1}\boldsymbol{F}_{tr}^\top\boldsymbol{R}_{tr}^{-1} + \boldsymbol{r}_{te}^\top\boldsymbol{R}_{tr}^{-1}\left(\boldsymbol{I}_{n_s} - \boldsymbol{F}_{tr}\boldsymbol{Q}^{-1}\boldsymbol{F}_{tr}^\top\right)\boldsymbol{R}_{tr}^{-1}.$$

Equivalently, it must be shown that

$$\boldsymbol{b}^\top\boldsymbol{R}_{tr}\boldsymbol{b} - 2\boldsymbol{b}^\top\boldsymbol{r}_{te} = -\boldsymbol{r}_{te}^\top\boldsymbol{R}_{tr}^{-1}\boldsymbol{r}_{te} + \boldsymbol{h}^\top\boldsymbol{Q}^{-1}\boldsymbol{h}. \tag{3.6.8}$$

Equality (3.6.8) can be proved using straightforward matrix algebra.

3.6.3 Implementation Issues

This section considers the choice of parameterization of the Gaussian correlation function. Recall the five parameterizations of the Gaussian correlation function,

$$R(h\,|\,\text{params}) = \exp\left\{-\xi\, h^2\right\} = \exp\left\{-(h/\theta)^2\right\} = \rho^{h^2} = \rho_\star^{4h^2} = \exp\left\{-10^\tau h^2\right\},$$

introduced in Sect. 2.2.2 where the valid values of the model parameters are $\xi > 0$, $\theta > 0$, $\rho \in (0, 1)$, $\rho_\star \in (0, 1)$, and $\tau \in (-\infty, +\infty)$. Note that optimizations involving the parameters of these correlation functions based on ξ or θ are bounded below over $(0, +\infty)$ and those that use ρ or $\rho_\star$ are over a bounded interval, while one involving τ is unbounded.

Papers that use Bayesian methodology tend to use the ρ or $\rho_\star$ parameterizations because these forms simplify the interpretation of the parameters and hence the task of specifying prior information about the parameters (Higdon et al. (2004, 2008)).

Papers that use empirical/plug-in methodology tend to use one of the other three parameterizations. Most problems that the three forms seek to navigate concern the optimization of the likelihood (3.3.6) or REML likelihood (3.3.9) for "active" inputs. To illustrate the differences implicit in using the parameterizations, suppose it is desired to estimate the correlation parameter for a function $y(x)$ whose $x \in [0, 1]$ is very active. The input x will be active only if the GP model for $y(x)$ has the feature that $Cor[Y(0), Y(1)]$ is quite small. For example, consider the five draws from zero mean, unit variance, and GP $Y(x)$, $x \in [0, 1]$, with

$$Cor\,[Y(0), Y(1)] = 0.0005$$

which are plotted in Fig. 3.21. Given adequate data, any reasonable estimator of $Cor[Y(0), Y(1)]$ should be small.

Likelihoods tend to have multiple local minima for parameterizations that result in estimated values near a boundary of the model parameter space and hence are numerically difficult to identify. MacDonald et al. (2015) illustrate this phenomenon with plots of the likelihood for several data sets with active inputs. Returning to the example introduced in the previous paragraph, a small amount of algebra shows that when $\rho = 0.0005$, then the equivalent values of the other parameters are $\xi = 7.60$, $\theta = 0.132$, $\rho_\star = 0.150$, and $\tau = 0.8809$. Notice that $\rho_\star = 0.150$ is more centrally located in $[0, 1]$ than is $\rho = 0.0005$ which makes this parameter "easier" to estimate. A similar statement is true for the τ parameterization. In sum, the parameterizations $R(h) = \rho_\star^{4h^2}$ and $R(h) = \exp\left\{-10^\tau h^2\right\}$ pull estimates of parameters correspond-

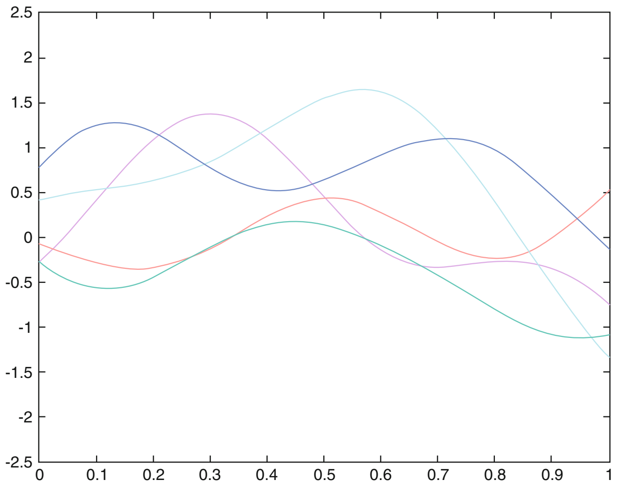


Fig. 3.21 Five draws from a stationary GP with zero mean, unit variance, and $Cor[Y(0), Y(1)] = 0.0005$

ing to active inputs toward the center of their parameter space allowing searches of local minima to be conducted more efficiently than for the correlation function $R(h) = \exp\left\{-\xi\, h^2\right\}$. Lastly, note that the factor 4 is not critical and can be replaced by another positive value depending on the anticipated activity of the inputs.

The issue of estimating the parameters associated with active inputs can be exacerbated when there are many inputs. For example, consider finding the MLE or REML of the correlation parameters when the product power exponential correlation model is assumed and there are $d = 20$ inputs. Because each input has unknown scale and power parameters, the determination of the MLE or REML of these parameters requires a 40-dimensional optimization. High-dimensional likelihood surfaces can have many local maxima, making global optimization difficult. Furthermore, typical algorithms used to perform such optimization require repeated evaluation of the determinant and inverse of the $n \times n$ correlation matrix $\boldsymbol{R}$. The Cholesky decomposition provides the most numerically stable method of calculating these quantities (Harville (1997)). Nevertheless, the repeated evaluation of the determinant and inverse of the correlation matrix at different correlation parameters is the most time-consuming aspect of algorithms that estimate correlation parameters. Indeed, methods that sequentially add data and update correlation parameter estimates must optimize an appropriate likelihood repeatedly.

As an example, Williams et al. (2000) report the times to maximize the REML likelihood which is required during the execution of their global optimization algorithm. In a six-dimensional input case, they fit the Matérn correlation function with a *single* shape parameter and separate range parameters for each input (a seven-dimensional $\boldsymbol{\kappa}$ correlation parameter). When 50 training points were used, their optimization of the $\boldsymbol{\kappa}$ likelihood (3.3.9) required 2140 s of Sun Ultra 5 CPU time, and this optimization required 4230 s of CPU time for 82 training points. Fitting the power exponential model was faster with 1105 s of CPU time required for the 50-point case and 3100 s of CPU time for the 82-point case. To ease the computational burden, applications that require a sequence of correlation parameter estimates for increasing n often re-estimate these parameters only periodically, for example, after every fifth point is added to the design. A more rational plan is to re-estimate the correlation parameters more often for small n when the estimates might be less stable and then less frequently for large n. For sufficiently large n, these estimators become intractable to calculate.

Many algorithmic approaches have been used successfully to estimate correlation parameters and expected improvement surfaces. Among these are the Nelder–Mead simplex algorithm (Nelder and Mead (1965)), branch and bound algorithms (Jones et al. (1998), Franey et al. (2011)), stochastic global optimization algorithms (Rinnooy Kan and Timmer (1984)), and quasi-Newton algorithms starting from one or more carefully selected initial points (e.g., Leatherman et al. (2014), MacDonald et al. (2015)). Appendix C summarizes several of these approaches.

As one specific example of an algorithm used to provide high-dimensional MLE and REML parameter estimation, Welch et al. (1992) proposed using a dimensionality reduction scheme to perform a series of presumably simpler optimizations. This approach is particularly useful in high-dimensional MLE and REML parameter estimation problems. The idea is to consider a sequence of optimizations by constraining the number of free parameters allowed in each optimization (minimization in the pseudo code below) where only "important" input variables are allowed to possess their own unconstrained correlation parameters. To illustrate, consider finding the REML of the correlation parameter $\boldsymbol{\kappa} = (\kappa_1, \ldots, \kappa_d)$ in some GP model. First, scale each of the d inputs to have the *same range*. At each stage of the process, let C denote the indices of the variables having *common* values of the correlation parameters for that step, and let $C \setminus \{j\}$ denote the set difference of C and $\{j\}$. In the following meta-code, *S0* is an initialization step, while *S1* and *S2* are induction steps.

```
S0 Set $C = \{1, 2, \ldots, d\}$, i.e., $\kappa_1 = \cdots = \kappa_d = \kappa$. Maximize (3.3.9) as a function of $\kappa$,
   and denote the resulting log likelihood by $\ell_0$.
S1 For each $j \in C$, maximize (3.3.9) under the constraint that variables $\kappa_h$ with
   $h \in C \setminus \{j\}$ have a common value and $\kappa_j$ varies freely. Denote the log likelihood
   evaluated at the estimated $\boldsymbol{\kappa}$ by $\ell_j$.
S2 Let $j^{\max}$ denote that $j \in C$ producing the largest increase in $\ell_j - \ell_0$.
S3 If $\ell_{j^{\max}} - \ell_0$ represents a "significant" increase in the log likelihood as judged by
   a stopping criterion, then update $C$ to be $C \setminus j^{\max}$, $\ell_0$ to be $\ell_{j^{\max}}$, and fix $\kappa_{j^{\max}}$ at its
   value estimated in S1. Continue the next iteration at step S1. Otherwise, stop the
   algorithm, and estimate the correlation parameters to be the values produced by
   the previous iteration.
```

Variations are, of course, possible in this and most basic algorithms. For example, two-dimensional optimizations are used in every cycle of *S1* because all $\kappa_{j^{\max}}$ estimated in previous cycles are fixed in subsequent ones. Instead, *S1* could allow the κ_j values previously estimated to vary freely along with the next individual κ_j to be estimated.

In sum, the primary feature of a successful algorithm for maximizing the likelihood or REML likelihood is that it must be capable of identifying a global maximum/minimum when the surface to be optimized has many local maxima/minima. Current proposals use a combination of local search Newton or quasi-Newton algorithms with starting values determined by some form of global search. In a recent article, Butler et al. (2014) give a survey of algorithms for MLE/REML estimation and a new proposal based on these two-stage ideas.

3.6.4 Software for Computing EBLUPs

The authors of this volume provide a partial list of software for fitting EBLUPs to the GP models described in this chapter, without endorsing any particular package. The programs below differ in:

- the correlation functions fit,
- the mean models that can be used,
- the correlation parameter estimation methods available,

- the UQ measures produced, and
- their prediction capabilities.

The list below is not a review nor a comparison of the accuracy and features of the programs. Most pieces of software are in continuous development and web searches will provide up-to-date information about the software as well as other packages that have been developed since the publication of this book.

1. DiceKriging (an R package),
2. CPErK (written in C and the unix scripting language),
3. DACE (a MATLAB program available at http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=1460),
4. Dakota (http://dakota.sandia.gov),
5. GaSP (available from W. Welch),
6. GPfit (an R package),
7. GPMfit (a MATLAB version of GPfit available at https://sourceforge.net/projects/gpmfit),
8. JMP,
9. SAS Proc Mixed,
10. MUCM website contains a list of Bayesian and Frequentist software for prediction and calibration (http://www.mucm.ac.uk/Pages/MCSGSoftware.html)

3.6.5 Alternatives to Kriging Metamodels and Other Topics

3.6.5.1 Alternatives to Kriging Metamodels

This chapter has discussed predicting outputs from computer experiments using kriging metamodels. Making use of the regression term in the model (3.1.1) allows one to fit "nonstationary" data by including *known* regression functions to describe long-term trends and a GP to describe small-scale deviations. The text has described a number of alternatives to estimating the model parameters for the kriging model.

Several competitors to the kriging model and its prediction have been proposed in the literature. The list below gives several options along with references.

- Polynomial chaos expansions (a polynomial response surface model). See, for example, Blatman and Sudret (2010).
- Neural networks. See, for example, Vicario et al. (2016).
- Radial basis functions and other basis functions. See, for example, Buhmann (2003) and Chakraborty et al. (2017).
- Splines. See, for example, Ben-Ari and Steinberg (2007), Reich et al. (2009), and Stripling et al. (2013).

More importantly, there have been a number of proposals for fitting adaptive, flexible emulators to data from $y(\boldsymbol{x})$ which are nonstationary. A selection of these approaches is sketched below.

- Gramacy and Lee (2008) describe a tree-based method of dividing the input space and fitting different GP models within each subdivision of the input space that has been identified (*treed Gaussian processes*).
- Joseph et al. (2008) introduce *blind kriging*, a method that relaxes the assumption that the regression functions $f_i(\boldsymbol{x})$ in (3.1.1) be known. Rather it selects regression functions from a candidate set using a variable selection method.
- Ba and Joseph (2012) also replace the regression model in (3.1.1). They propose using an independent GP to describe large-scale behavior of $y(\boldsymbol{x})$ over different parts of the input space and a second GP to describe small-scale deviations.
- Davis (2015) develops a Bayesian prior for the composite GP of Ba and Joseph (2012), an MCMC algorithm for implementing the model, and shows its usefulness in applications such as variable screening.

3.6.5.2 Testing the Covariance Structure

Although this topic has not been discussed, there are methods for testing the appropriateness of some aspects of the covariance structure of GP models. References include Mitchell et al. (2005), Li et al. (2008), and Bastos and O'Hagan (2009).

Chapter 4
Bayesian Inference for Simulator Output

4.1 Introduction

In Chap. 3 the correlation and precision parameters are completely unknown for the process model assumed to generate simulator output. In contrast this chapter assumes that the researcher has prior knowledge about the unknown parameters that is quantifiable in the form of a prior distribution. The source of the prior knowledge is usually a combination of expert opinion and previous experience with data from similar physical systems. The prior is simplest to determine if the parameters are selected to be interpretable quantities for the simulator output, e.g., parameters that govern the "overall" mean output, its range, or the number of local maxima and minima all would be amenable to elicit prior knowledge.

The following notation is used to state the objectives of this chapter. Let $y(\boldsymbol{x})$ denote the simulator output; let $\boldsymbol{y}^{tr} = (y(\boldsymbol{x}_1^{tr}), \ldots, y(\boldsymbol{x}_{n_s}^{tr}))^\top$ denote the (known) simulator outputs evaluated at the n_s "training" inputs $\boldsymbol{x}_1^{tr}, \ldots, \boldsymbol{x}_{n_s}^{tr}$; and let $\boldsymbol{y}^{te} = (y(\boldsymbol{x}_1^{te}), \ldots, y(\boldsymbol{x}_{n_e}^{te}))^\top$ denote the (unknown) simulator outputs which are to be predicted at n_e "test" inputs $\boldsymbol{x}_1^{te}, \ldots, \boldsymbol{x}_{n_e}^{te}$. Finally, let $\boldsymbol{Y}^{te} = (Y(\boldsymbol{x}_1^{te}), \ldots, Y(\boldsymbol{x}_{n_e}^{te}))^\top$ and $\boldsymbol{Y}^{tr} = (Y(\boldsymbol{x}_1^{tr}), \ldots, Y(\boldsymbol{x}_{n_s}^{tr}))^\top$ denote the process models for the test and training data, respectively.

The Bayesian models used in this chapter are two-stage hierarchical models. Here, the top-most stage specifies a conditional distribution for $(\boldsymbol{Y}^{te}, \boldsymbol{Y}^{tr})$ given the model parameters. Then a distribution for the unknown parameters is specified, using the ideas mentioned in the first paragraph of this section.

Specifically, the top stage of the Bayesian model used in this chapter assumes that $(\boldsymbol{y}^{te}, \boldsymbol{y}^{tr})$ can be viewed as a draw from a regression + stationary Gaussian process (GP) model

$$\left[\begin{pmatrix} \boldsymbol{Y}^{te} \\ \boldsymbol{Y}^{tr} \end{pmatrix} \middle| \, \boldsymbol{\vartheta} \right] \sim N_{n_e+n_s} \left(\begin{pmatrix} \boldsymbol{F}_{te} \\ \boldsymbol{F}_{tr} \end{pmatrix} \boldsymbol{\beta}, \; \lambda_Z^{-1} \begin{bmatrix} \boldsymbol{R}_{te} & \boldsymbol{R}_{te,tr} \\ \boldsymbol{R}_{te,tr}^\top & \boldsymbol{R}_{tr} \end{bmatrix} \right) \tag{4.1.1}$$

given the *unknown* model parameters $\boldsymbol{\vartheta}$. In Sect. 4.2.1, the parameter $\boldsymbol{\vartheta}$ is $\boldsymbol{\beta}$; in Sect. 4.2.2, $\boldsymbol{\vartheta}$ is $(\boldsymbol{\beta}, \lambda_Z)$; and in Sect. 4.3, $\boldsymbol{\vartheta}$ is $(\boldsymbol{\beta}, \lambda_Z, \boldsymbol{\kappa})$.

A second stage of the Bayesian model specifies the information that is known about $\boldsymbol{\vartheta}$ in the form of a prior distribution, denoted $[\boldsymbol{\vartheta}]$. Inference about the unknown model parameters $\boldsymbol{\vartheta}$ can be obtained from the conditional distribution of $\boldsymbol{\vartheta}$ given the training data, i.e., from the posterior distribution $[\boldsymbol{\vartheta} \mid \boldsymbol{Y}^{tr}]$. For example, the mean and standard deviation of $[\boldsymbol{\vartheta} \mid \boldsymbol{Y}^{tr}]$ are, respectively, the Bayesian estimate of $\boldsymbol{\vartheta}$ and a quantification of the uncertainty of the estimate. Similarly, the *predictive distribution* of $\boldsymbol{Y}^{te}$, defined to be $[\boldsymbol{Y}^{te} \mid \boldsymbol{Y}^{tr}]$, captures the information about $\boldsymbol{Y}^{te}$ that is contained in $\boldsymbol{Y}^{tr}$. For example, when (4.1.1) holds it is a fundamental result in multivariate analysis that, given $\boldsymbol{Y}^{tr}$ and $\boldsymbol{\vartheta}$,

$$\left[\boldsymbol{Y}^{te} \,\middle|\, \boldsymbol{Y}^{tr} = \boldsymbol{y}^{tr}, \boldsymbol{\vartheta}\right] \\ \sim N_{n_e}\left(\boldsymbol{F}_{te}\boldsymbol{\beta} + \boldsymbol{R}_{te,tr}\boldsymbol{R}_{tr}^{-1}\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\boldsymbol{\beta}\right), \lambda_Z^{-1}\left(\boldsymbol{R}_{te} - \boldsymbol{R}_{te,tr}\boldsymbol{R}_{tr}^{-1}\boldsymbol{R}_{te,tr}^{\top}\right)\right)$$

(see Lemma B.2). The density of $[\boldsymbol{Y}^{te} \mid \boldsymbol{Y}^{tr} = \boldsymbol{y}^{tr}]$, denoted $\pi(\boldsymbol{y}^{te} \mid \boldsymbol{y}^{tr})$, can be calculated using

$$\pi\left(\boldsymbol{y}^{te} \mid \boldsymbol{y}^{tr}\right) = \int \pi\left(\boldsymbol{y}^{te}, \boldsymbol{\vartheta} \mid \boldsymbol{y}^{tr}\right)\, d\boldsymbol{\vartheta} = \int \pi\left(\boldsymbol{y}^{te} \mid \boldsymbol{\vartheta}, \boldsymbol{y}^{tr}\right)\, \pi\left(\boldsymbol{\vartheta} \mid \boldsymbol{y}^{tr}\right)\, d\boldsymbol{\vartheta} \tag{4.1.2}$$

where $\pi(\boldsymbol{y}^{te}, \boldsymbol{\vartheta} \mid \boldsymbol{y}^{tr})$ and $\pi(\boldsymbol{\vartheta} \mid \boldsymbol{y}^{tr})$ are the density functions of $[\boldsymbol{Y}^{te}, \boldsymbol{\vartheta} \mid \boldsymbol{Y}^{tr} = \boldsymbol{y}^{tr}]$ and $[\boldsymbol{\vartheta} \mid \boldsymbol{Y}^{tr} = \boldsymbol{y}^{tr}]$, respectively. As above, the mean of the predictive distribution,

$$\widehat{\boldsymbol{y}}^{te} = E\left[\boldsymbol{Y}^{te} \,\middle|\, \boldsymbol{Y}^{tr}\right],$$

is the Bayes estimator of $\boldsymbol{y}^{te}$, while the diagonal elements of $Cov[\boldsymbol{Y}^{te} \mid \boldsymbol{Y}^{tr}]$ quantify the predictive uncertainty in each component of $\boldsymbol{y}^{te}$. Unfortunately in most practical applications of Bayesian methodology, the predictive $[\boldsymbol{Y}^{te} \mid \boldsymbol{Y}^{tr}]$ can only be sampled. In such cases, the mean, $\widehat{\boldsymbol{y}}^{te}$, and covariance $Cov[\boldsymbol{Y}^{te} \mid \boldsymbol{Y}^{tr}]$ are estimated from the samples drawn from $[\boldsymbol{Y}^{te} \mid \boldsymbol{Y}^{tr}]$.

To facilitate reading this chapter, the following notation used throughout is summarized next.

- $[\boldsymbol{W}]$ denotes the distribution of $\boldsymbol{W}$ (where needed, $\pi(\boldsymbol{w})$, $E[\boldsymbol{W}]$, and $Cov[\boldsymbol{W}]$ denote the (joint) probability density function of $\boldsymbol{W}$, the mean of $\boldsymbol{W}$, and the variance-covariance matrix of $\boldsymbol{W}$, respectively);
- $\boldsymbol{F}_{te}$ is the $n_e \times p$ matrix whose i^{th} row consists of the (known) regression functions for the input $\boldsymbol{x}_i^{te}$, $i = 1, \ldots, n_e$;
- $\boldsymbol{F}_{tr}$ is the $n_s \times p$ matrix of known regression functions for the n_s training data inputs;
- $\boldsymbol{\beta}$ denotes a $p \times 1$ vector of regression coefficients: $\boldsymbol{\beta}$ is *unknown* for all the models considered in this chapter;
- $\boldsymbol{\kappa}$ denotes the vector of parameters that determine a given correlation function: $\boldsymbol{\kappa}$ can be known or unknown depending on the model considered;
- $\boldsymbol{R}_{te}$ is the $n_e \times n_e$ correlation matrix $Cor[\boldsymbol{Y}^{te}, \boldsymbol{Y}^{te}]$;

- $\boldsymbol{R}_{te,tr}$ is the $n_e \times n_s$ cross-correlation matrix $Cor[\boldsymbol{Y}^{te}, \boldsymbol{Y}^{tr}]$;
- $\boldsymbol{R}_{tr}$ is the $n_s \times n_s$ correlation matrix $Cor[\boldsymbol{Y}^{tr}, \boldsymbol{Y}^{tr}]$;
- $\boldsymbol{\vartheta}$ denotes the vector of *all unknown* parameters for the model under discussion;
- λ_Z (λ_Z^{-1}) denotes the precision (variance) of the GP that describes deviations from the regression.

The chapter is organized as follows. Section 4.2 presents the densities and moments of $[\boldsymbol{\vartheta} \mid \boldsymbol{Y}^{tr}]$ and $[\boldsymbol{Y}^{te} \mid \boldsymbol{Y}^{tr}]$ for so-called "conjugate" cases which are $\vartheta = \boldsymbol{\beta}$ and $\boldsymbol{\vartheta} = (\boldsymbol{\beta}, \lambda_Z)$ in this chapter. A model is called conjugate if the prior and posterior distributions of $[\boldsymbol{\vartheta}]$ come from the same parametric family. Analytic expressions for the densities of posterior quantities can be given for conjugate models. Unfortunately the situations $\boldsymbol{\vartheta} = \boldsymbol{\beta}$ and $\boldsymbol{\vartheta} = (\boldsymbol{\beta}, \lambda_Z)$ are usually not directly useful in applications. Nevertheless, the posterior for the $\boldsymbol{\vartheta} = \boldsymbol{\beta}$ setting is straightforward to derive and will be given in detail to provide the reader with a sense of the analysis for more complicated cases. Section 4.3 presents posterior results when $\boldsymbol{\vartheta} = (\boldsymbol{\beta}, \lambda_Z, \boldsymbol{\kappa})$, which is useful in practical settings.

4.2 Inference for Conjugate Bayesian Models

4.2.1 Posterior Inference for Model (4.1.1) When $\boldsymbol{\vartheta} = \boldsymbol{\beta}$

Assume that the training and test data can be described as draws from the regression + stationary GP model (4.1.1) in which the regression coefficient is *unknown* but the process precision and correlations are *known*, i.e., $\boldsymbol{\vartheta} = \boldsymbol{\beta}$. Theorem 4.1 provides the posterior parameter distribution $[\boldsymbol{\beta} \mid \boldsymbol{Y}^{tr}]$ and predictive distribution $[\boldsymbol{Y}^{te} \mid \boldsymbol{Y}^{tr}]$ for two choices of second-stage $\boldsymbol{\beta}$ priors. The first prior, denoted Case (a), is a multivariate normal prior which can be regarded as an informative choice. The second prior, Case (b), can be thought of as a non-informative prior. The non-informative prior can be regarded as the limit of normal priors in which the precision tends to zero. More formally the predictive distribution for Case (b) is obtained by letting the prior precision $\lambda_\beta \to 0$ in the predictive distribution for prior (a). The Bayesian predictor for the test data given the training data, $E[\boldsymbol{Y}^{te} \mid \boldsymbol{Y}^{tr}]$, and the uncertainty quantification $Cov[\boldsymbol{Y}^{te} \mid \boldsymbol{Y}^{tr}]$ have closed forms for both priors.

Theorem 4.1. Suppose $(\boldsymbol{Y}^{te}, \boldsymbol{Y}^{tr})$ follows a two-stage hierarchical model with top stage

$$\left[\begin{pmatrix} \boldsymbol{Y}^{te} \\ \boldsymbol{Y}^{tr} \end{pmatrix} \middle| \, \boldsymbol{\beta} \right] \sim N_{n_e+n_s}\left(\begin{pmatrix} \boldsymbol{F}_{te} \\ \boldsymbol{F}_{tr} \end{pmatrix} \boldsymbol{\beta},\, \lambda_Z^{-1} \begin{bmatrix} \boldsymbol{R}_{te} & \boldsymbol{R}_{te,tr} \\ \boldsymbol{R}_{te,tr}^\top & \boldsymbol{R}_{tr} \end{bmatrix} \right),$$

where $\boldsymbol{\beta}$ is *unknown* while λ_Z and all correlations are *known*.

(a) Suppose that

$$[\boldsymbol{\beta}] \sim N_p\left(\boldsymbol{b}_\beta, \lambda_\beta^{-1} \boldsymbol{V}_\beta\right)$$

is the second-stage model with (known) prior parameter $(\boldsymbol{V}_\beta, \boldsymbol{b}_\beta, \lambda_\beta)$, where $\boldsymbol{V}_\beta$ is a positive definite matrix, $\boldsymbol{b}_\beta \in \mathbb{R}^p$, and $\lambda_\beta > 0$. Then the posterior distribution of $\boldsymbol{\beta}$ is

$$\left[\boldsymbol{\beta} \mid \boldsymbol{Y}^{tr} = \boldsymbol{y}^{tr}\right] \sim N_p\left(\boldsymbol{\mu}_{\beta|tr}, \boldsymbol{\Sigma}_{\beta|tr}\right),$$

where

$$\boldsymbol{\mu}_{\beta|tr} = \left(\lambda_Z\, \boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1} \boldsymbol{F}_{tr} + \lambda_\beta\, \boldsymbol{V}_\beta^{-1}\right)^{-1} \times \left(\lambda_Z\, \boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1} \boldsymbol{y}^{tr} + \lambda_\beta\, \boldsymbol{V}_\beta^{-1} \boldsymbol{b}_\beta\right), \tag{4.2.1}$$

and

$$\boldsymbol{\Sigma}_{\beta|tr} = \left(\lambda_Z \boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1} \boldsymbol{F}_{tr} + \lambda_\beta \boldsymbol{V}_\beta^{-1}\right)^{-1}. \tag{4.2.2}$$

The predictive distribution of $\boldsymbol{Y}^{te}$ is

$$\left[\boldsymbol{Y}^{te} \mid \boldsymbol{Y}^{tr} = \boldsymbol{y}^{tr}\right] \sim N_{n_e}\left(\boldsymbol{\mu}_{te|tr}, \boldsymbol{\Sigma}_{te|tr}\right), \tag{4.2.3}$$

with mean (vector)

$$\boldsymbol{\mu}_{te|tr} = \boldsymbol{F}_{te} \boldsymbol{\mu}_{\beta|tr} + \boldsymbol{R}_{te,tr} \boldsymbol{R}_{tr}^{-1}\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr} \boldsymbol{\mu}_{\beta|tr}\right), \tag{4.2.4}$$

and covariance matrix

$$\boldsymbol{\Sigma}_{te|tr} = \lambda_Z^{-1}\left\{\boldsymbol{R}_{te} - \left(\boldsymbol{F}_{te}\ \boldsymbol{R}_{te,tr}\right)\begin{bmatrix} -\frac{\lambda_\beta}{\lambda_Z}\boldsymbol{V}_\beta^{-1} & \boldsymbol{F}_{tr}^\top \\ \boldsymbol{F}_{tr} & \boldsymbol{R}_{tr} \end{bmatrix}^{-1} \begin{pmatrix} \boldsymbol{F}_{te}^\top \\ \boldsymbol{R}_{te,tr}^\top \end{pmatrix}\right\}. \tag{4.2.5}$$

(b) Suppose that

$$\pi(\boldsymbol{\beta}) \propto 1$$

on $\mathbb{R}^p$, then the posterior distribution of $\boldsymbol{\beta}$ is

$$\left[\boldsymbol{\beta} \mid \boldsymbol{Y}^{tr} = \boldsymbol{y}^{tr}\right] \sim N_p\left(\widehat{\boldsymbol{\beta}} \equiv \left(\boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1} \boldsymbol{F}_{tr}\right)^{-1} \times \left(\boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1} \boldsymbol{y}^{tr}\right), \lambda_Z^{-1}\left(\boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1} \boldsymbol{F}_{tr}\right)^{-1}\right).$$

The predictive distribution of $\boldsymbol{Y}^{te}$ is

$$\left[\boldsymbol{Y}^{te} \mid \boldsymbol{Y}^{tr} = \boldsymbol{y}^{tr}\right] \sim N_{n_e}\left(\boldsymbol{\mu}_{te|tr}, \boldsymbol{\Sigma}_{te|tr}\right), \tag{4.2.6}$$

where the mean $\boldsymbol{\mu}_{te|tr}$ is the modification of (4.2.4) which replaces $\boldsymbol{\mu}_{\beta|tr}$ by $\widehat{\boldsymbol{\beta}}$ and the covariance $\boldsymbol{\Sigma}_{te|tr}$ is the modification of (4.2.5) which replaces $\frac{\lambda_\beta}{\lambda_Z}\boldsymbol{V}_\beta^{-1}$ by the $p \times p$ matrix of zeros.

The proof of Theorem 4.1 is given in Sect. 4.4. It requires straightforward calculations to implement the right-hand integral in the general strategy of (4.1.2).

4.2.1.1 Posterior Inference About $\boldsymbol{\beta}$

Consider inferences about $\boldsymbol{\beta}$ that are provided by the mean and covariance matrix of $[\boldsymbol{\beta} \mid \boldsymbol{Y}^{tr}]$ under prior (a). The posterior mean of $\boldsymbol{\beta}$ depends only on the ratio λ_β/λ_Z because

$$\begin{aligned}\boldsymbol{\mu}_{\beta|tr} &= \lambda_Z^{-1}\left(\boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr} + \boldsymbol{V}_\beta^{-1}\,\lambda_\beta/\lambda_Z\right)^{-1} \times \lambda_Z\left(\boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{y}^{tr} + \boldsymbol{V}_\beta^{-1}\boldsymbol{b}_\beta\,\lambda_\beta/\lambda_Z\right)\\ &= \left(\boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr} + \boldsymbol{V}_\beta^{-1}\,\lambda_\beta/\lambda_Z\right)^{-1} \times \left(\boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}} + \boldsymbol{V}_\beta^{-1}\boldsymbol{b}_\beta\,\lambda_\beta/\lambda_Z\right).\end{aligned}$$

When the prior precision parameter of $\boldsymbol{\beta}$ is equal to the process precision of $Y(\boldsymbol{x})$, i.e., $\lambda_\beta = \lambda_Z$, the posterior mean simplifies further to

$$\begin{aligned}\boldsymbol{\mu}_{\beta|tr} &= \left(\boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr} + \boldsymbol{V}_\beta^{-1}\right)^{-1} \times \left(\boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}} + \boldsymbol{V}_\beta^{-1}\boldsymbol{b}_\beta\right)\\ &= \boldsymbol{\Omega}\,\widehat{\boldsymbol{\beta}} + (\boldsymbol{I}_p - \boldsymbol{\Omega})\,\boldsymbol{b}_\beta \qquad (4.2.7)\end{aligned}$$

where $\boldsymbol{\Omega} = \left(\boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr} + \boldsymbol{V}_\beta^{-1}\right)^{-1}\left(\boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr}\right)$, which shows that the posterior mean is a matrix "convex combination" of the BLUP of $\boldsymbol{\beta}$, $\widehat{\boldsymbol{\beta}}$, and its prior mean, $\boldsymbol{b}_\beta$. In contrast, the posterior covariance,

$$\boldsymbol{\Sigma}_{\beta|tr} = \left[\lambda_Z \boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr} + \lambda_\beta \boldsymbol{V}_\beta^{-1}\right]^{-1} = \lambda_Z^{-1}\left[\boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr} + \boldsymbol{V}_\beta^{-1}\lambda_\beta/\lambda_Z\right]^{-1},$$

depends on both λ_Z and λ_β or equivalently on λ_Z and λ_β/λ_Z (which has been noted that the ratio = 1 when $\lambda_\beta = \lambda_Z$).

4.2.1.2 Predictive Inference at a Single Test Input $\boldsymbol{x}^{te}$

To provide additional insight about the nature of the Bayesian predictor, consider the case of a *single test input*. Throughout this discussion, the predictive mean is regarded as a function of random training data $\boldsymbol{Y}^{tr}$. For both priors (a) and (b) the predictive mean is linear in $\boldsymbol{Y}^{tr}$, is an unbiased predictor of $Y(\boldsymbol{x}^{te})$, i.e., the predictive mean equals the mean of $Y(\boldsymbol{x}^{te})$, and it interpolates the training data.

Consider prior (a) and let $\boldsymbol{x}^{te}$ denote the test input, $\mu_{te|tr}(\boldsymbol{x}^{te})$ the Bayes predictor in (4.2.4), and $\sigma^2_{te|tr}(\boldsymbol{x}^{te})$ the real-valued version of the posterior covariance $\boldsymbol{\Sigma}_{te|tr}$ in (4.2.5). Algebra shows that $\mu_{te|tr}(\boldsymbol{x}^{te})$ is linear in $\boldsymbol{Y}^{tr}$ and, with additional calculation, that it is an unbiased predictor of $Y(\boldsymbol{x}^{te})$, i.e., $\mu_{te|tr}(\boldsymbol{x}^{te})$ has the same mean as $Y(\boldsymbol{x}^{te})$. Continuity and other smoothness properties of $\mu_{te|tr}(\boldsymbol{x}^{te})$ are inherited from those of the correlation function $R(\cdot)$ and the regressors $\{f_j(\cdot)\}_{j=1}^p$ because

$$\begin{aligned}\mu_{te|tr}(\boldsymbol{x}^{te}) &= \boldsymbol{f}^\top(\boldsymbol{x}^{te})\boldsymbol{\mu}_{\beta|tr} + \boldsymbol{r}_{te}^\top \boldsymbol{R}_{tr}^{-1}(\boldsymbol{Y}^{tr} - \boldsymbol{F}_{tr}\boldsymbol{\mu}_{\beta|tr})\\ &= \sum_{j=1}^{p} f_j(\boldsymbol{x}^{te})\,\mu_{\beta|tr,j} + \sum_{i=1}^{n_s} d_i\,R(\boldsymbol{x}^{te} - \boldsymbol{x}_i^{tr}),\end{aligned}$$

where $\mu_{\beta|tr,j}$ is the j^{th} element of $\boldsymbol{\mu}_{\beta|tr}$ and d_i is the i^{th} element of the $n_s \times 1$ vector $\boldsymbol{R}_{tr}^{-1}(\boldsymbol{Y}^{tr} - \boldsymbol{F}_{tr}\boldsymbol{\mu}_{\beta|tr})$. Previously, Sect. 3.3.1 had observed linearity and unbiasedness of the BLUP (3.2.7) which is the posterior mean for prior (b). Lastly, the predictive mean $\mu_{te|tr}(\boldsymbol{x}^{te})$ interpolates the training data. For prior (a) this is true because when $\boldsymbol{x}^{te} = \boldsymbol{x}_i^{tr}$ for a given $i \in \{1, \ldots, n_s\}$, $\boldsymbol{f}(\boldsymbol{x}^{te}) = \boldsymbol{f}(\boldsymbol{x}_i^{tr})$, and $\boldsymbol{r}_{te}^{\top}\boldsymbol{R}_{tr}^{-1} = \boldsymbol{e}_i^{\top}$, the i^{th} unit vector. Thus

$$\begin{aligned}
\mu_{te|tr}(\boldsymbol{x}^{te}) &= \boldsymbol{f}^{\top}(\boldsymbol{x}_i^{tr})\,\boldsymbol{\mu}_{\beta|tr} + \boldsymbol{r}_{te}^{\top}\boldsymbol{R}_{tr}^{-1}(\boldsymbol{Y}^{tr} - \boldsymbol{F}_{tr}\boldsymbol{\mu}_{\beta|tr}) \\
&= \boldsymbol{f}^{\top}(\boldsymbol{x}_i^{tr})\,\boldsymbol{\mu}_{\beta|tr} + \boldsymbol{e}_i^{\top}(\boldsymbol{Y}^{tr} - \boldsymbol{F}_{tr}\,\boldsymbol{\mu}_{\beta|tr}) \\
&= \boldsymbol{f}^{\top}(\boldsymbol{x}_i^{tr})\,\boldsymbol{\mu}_{\beta|tr} + (Y_i - \boldsymbol{f}^{\top}(\boldsymbol{x}_i^{tr})\boldsymbol{\mu}_{\beta|tr}) \\
&= Y_i\,.
\end{aligned}$$

A similar argument holds for the predictive mean for prior (b).

For prior (b) the posterior variance $\sigma^2_{te|tr}(\boldsymbol{x}^{te})$ simplifies to

$$\sigma^2_{te|tr}(\boldsymbol{x}^{te}) = \lambda_Z^{-1}\left\{1 - \boldsymbol{r}_{te}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{r}_{te} + \boldsymbol{h}^{\top}(\boldsymbol{F}_{tr}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr})^{-1}\boldsymbol{h}\right\} \tag{4.2.8}$$

where $\boldsymbol{h} = \boldsymbol{f}_{te} - \boldsymbol{F}_{tr}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{r}_{te}$. Equation (4.2.8) was given previously as the variance of the BLUP (3.2.7). For prior (a), the posterior variance has a form similar to (4.2.8) with one modification. Using matrix identities and algebra shows

$$\begin{aligned}
\sigma^2_{te|tr}(\boldsymbol{x}^{te}) &= \lambda_Z^{-1}\left\{1 - \begin{pmatrix}\boldsymbol{f}_{te}^{\top} & \boldsymbol{r}_{te}^{\top}\end{pmatrix}\begin{bmatrix}-\frac{\lambda_\beta}{\lambda_Z}\boldsymbol{V}_\beta^{-1} & \boldsymbol{F}_{tr}^{\top} \\ \boldsymbol{F}_{tr} & \boldsymbol{R}_{tr}\end{bmatrix}^{-1}\begin{pmatrix}\boldsymbol{f}_{te} \\ \boldsymbol{r}_{te}\end{pmatrix}\right\} \\
&= \lambda_Z^{-1}\Big\{1 - \big[-\boldsymbol{f}_{te}^{\top}\boldsymbol{Q}^{-1}\boldsymbol{f}_{te} + 2\boldsymbol{f}_{te}^{\top}\boldsymbol{Q}^{-1}\boldsymbol{F}_{tr}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{r}_{te} \\
&\qquad + \ \boldsymbol{r}_{te}^{\top}\{\boldsymbol{R}_{tr}^{-1} - \boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr}\boldsymbol{Q}^{-1}\boldsymbol{F}_{tr}^{\top}\boldsymbol{R}_{tr}^{-1}\}\boldsymbol{r}_{te}\big]\Big\} &\quad (4.2.9)\\
&= \lambda_Z^{-1}\Big\{1 - \boldsymbol{r}_{te}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{r}_{te} + \boldsymbol{f}_{te}^{\top}\boldsymbol{Q}^{-1}\boldsymbol{f}_{te} - 2\boldsymbol{f}_{te}^{\top}\boldsymbol{Q}^{-1}\boldsymbol{F}_{tr}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{r}_{te} \\
&\qquad + \boldsymbol{r}_{te}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr}\boldsymbol{Q}^{-1}\boldsymbol{F}_{tr}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{r}_{te}\Big\} \\
&= \lambda_Z^{-1}\left\{1 - \boldsymbol{r}_{te}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{r}_{te} + \boldsymbol{h}^{\top}\boldsymbol{Q}^{-1}\boldsymbol{h}\right\}, &\quad (4.2.10)
\end{aligned}$$

where $\boldsymbol{h} = \boldsymbol{f}_{te} - \boldsymbol{F}_{tr}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{r}_{te}$ and

$$\boldsymbol{Q} = \boldsymbol{F}_{tr}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr} + \frac{\lambda_\beta}{\lambda_Z}\boldsymbol{V}_\beta^{-1}\,; \tag{4.2.11}$$

the equality in (4.2.9) follows from Lemma B.3.

Intuition suggests that the posterior variance of $Y(\boldsymbol{x}^{te})$ given the training data $\boldsymbol{Y}^{tr}$ should be *zero* whenever $\boldsymbol{x}^{te} = \boldsymbol{x}_i^{tr}$, $1 \le i \le n_s$ because $y(\boldsymbol{x}_i^{tr})$ is *known* exactly at all the training data sites and there is no measurement error term in the stochastic process model. Recall that $\sigma^2_{te|tr}(\boldsymbol{x}^{te}) = 0$ whenever $\boldsymbol{x}^{te}$ is a training input was shown previously for (4.2.8), the posterior variance under prior (b). To see that $\sigma^2_{te|tr}(\boldsymbol{x}^{te}) = 0$ also holds for prior (a) when $\boldsymbol{x}^{te}$ is a training input, fix $\boldsymbol{x}^{te} = \boldsymbol{x}_1^{tr}$ to simplify notation. In this case, recall that $\boldsymbol{r}_{te}^\top \boldsymbol{R}_{tr}^{-1} = \boldsymbol{e}_1^\top$, and observe that $\boldsymbol{f}_{te} = \boldsymbol{f}(\boldsymbol{x}^{te}) = \boldsymbol{f}(\boldsymbol{x}_1^{tr})$. From (4.2.10) and recalling $\boldsymbol{Q}$ is given by (4.2.11),

$$\begin{aligned}
\sigma^2_{te|tr}(\boldsymbol{x}_1^{tr}) &= \lambda_Z^{-1}\left\{1 - \boldsymbol{r}_{te}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{r}_{te} + (\boldsymbol{f}^\top(\boldsymbol{x}_1^{tr}) - \boldsymbol{r}_{te}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr})\boldsymbol{Q}^{-1}(\boldsymbol{f}(\boldsymbol{x}_1^{tr}) - \boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{r}_{te})\right\} \\
&= \lambda_Z^{-1}\left\{1 - \boldsymbol{e}_1^\top \boldsymbol{r}_{te}(\boldsymbol{x}_1^{tr}) + (\boldsymbol{f}^\top(\boldsymbol{x}_1^{tr}) - \boldsymbol{e}_1^\top \boldsymbol{F}_{tr})\boldsymbol{Q}^{-1}(\boldsymbol{f}(\boldsymbol{x}_1^{tr}) - \boldsymbol{F}_{tr}^\top \boldsymbol{e}_1)\right\} \\
&= \lambda_Z^{-1}\left\{1 - 1 + (\boldsymbol{f}^\top(\boldsymbol{x}_1^{tr}) - \boldsymbol{f}^\top(\boldsymbol{x}_1^{tr}))\boldsymbol{Q}^{-1}(\boldsymbol{f}(\boldsymbol{x}_1^{tr}) - \boldsymbol{f}(\boldsymbol{x}_1^{tr}))\right\} \\
&= \lambda_Z^{-1}\{1 - 1 + 0\} = 0.
\end{aligned}$$

Perhaps the most important use of (4.2.3) or (4.2.6) in Theorem 4.1 is to provide pointwise predictive uncertainty bands for $y(\boldsymbol{x}^{te})$ by using the fact that conditionally,

$$\frac{Y(\boldsymbol{x}^{te}) - \mu_{te|tr}(\boldsymbol{x}^{te})}{\sigma_{te|tr}(\boldsymbol{x}^{te})} \sim N(0,1)\,. \tag{4.2.12}$$

Equation (4.2.12) gives the posterior prediction interval

$$P\left\{ Y(\boldsymbol{x}^{te}) \in \mu_{te|tr}(\boldsymbol{x}^{te}) \pm \sigma_{te|tr}(\boldsymbol{x}^{te})\, z^{\alpha/2} \;\middle|\; \boldsymbol{Y}^{tr}\right\} = 1 - \alpha,$$

where $z^{\alpha/2}$ is the upper $\alpha/2$ critical point of the standard normal distribution (see Appendix A). As a special case suppose $x^{te} \in (a, b)$, then

$$\mu_{te|tr}(x^{te}) \pm \sigma_{te|tr}(x^{te})\, z^{\alpha/2}$$

are pointwise $100(1-\alpha)\%$ prediction bands for $y(x^{te})$, $a < x^{te} < b$. The prediction band calculation is illustrated in the following example.

Example 4.1 (Damped Sine Curve). This example illustrates the effect of the prior $[\boldsymbol{\beta}]$ on the mean of the predictive distribution $\mu_{te|tr}(\boldsymbol{x}^{te})$ in Theorem 4.1. Consider the damped cosine function

$$y(x) = e^{-1.4x}\cos(7\pi x/2)\,, \qquad 0 < x < 1,$$

which is shown as the solid curve in Fig. 4.1. The figure also shows training data taken at $n_s = 7$ values which are shown as filled circles.

For any $x^{te} \in (0, 1)$, the predictive distribution of $Y(x^{te})$ is based on the hierarchical Bayes model whose first stage is the stationary stochastic process

$$[Y(x)\,|\,\beta_0] = \beta_0 + Z(x)\,, \qquad 0 < x < 1,$$

where $\beta_0 \in \mathbb{R}$ is unknown; the correlation function governing $Z(x)$ is taken to be $R(h) = \exp\{-10.0 \times h^2\}$.

To apply prior (a) of Theorem 4.1, assume that $\beta_0 \sim N(b_{te}, (\lambda_\beta)^{-1} \times v_{te}^2)$ where $v_{te} = 1$, b_{te} is the known prior mean, and λ_β is the known prior precision. For any $x^{te} \in (0, 1)$, the Bayesian predictor of $y(x^{te})$ given $\boldsymbol{Y}^{tr} = \boldsymbol{y}^{tr}$ is the posterior mean (4.2.4) which reduces to

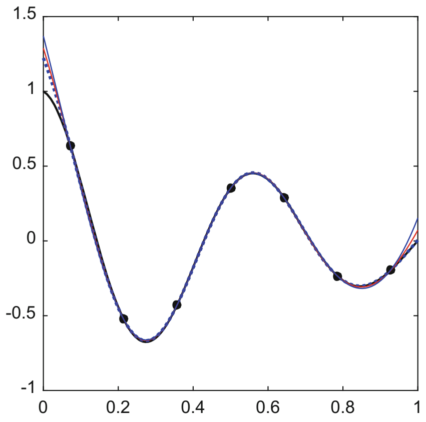


Fig. 4.1 The function $y(x) = \exp\{-1.4x\} \times \cos(3.5\pi x)$ (solid curve); a seven-point training data set (filled circles); the Bayesian predictor $\mu_{te|tr} = \mu_{\beta|tr} + \boldsymbol{r}_{te}^\top \boldsymbol{R}_{tr}^{-1}(\boldsymbol{y}^{tr} - \boldsymbol{1}_{n_s}\mu_{\beta|tr})$ in (4.2.13) and (4.2.14) for $\lambda_\beta = 0$ (blue), for $\lambda_\beta = 10$ (red), and for $\lambda_\beta = 100$ (green)

$$\mu_{te|tr}(x^{te}) = \mu_{\beta|tr} + \boldsymbol{r}_{te}^\top \boldsymbol{R}_{tr}^{-1}\left(\boldsymbol{y}^{tr} - \boldsymbol{1}_{n_s}\mu_{\beta|tr}\right), \tag{4.2.13}$$

where $\mu_{\beta|tr}$ is the posterior mean of β_0 given $\boldsymbol{y}^{tr}$,

$$\begin{aligned}\mu_{\beta|tr} &= \frac{\left(\boldsymbol{1}_{n_s}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{y}^{tr} + b_{te}\lambda_\beta/\lambda_Z\right)}{\left(\boldsymbol{1}_{n_s}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{1}_{n_s} + \lambda_\beta/\lambda_Z\right)} \\ &= \omega\, b_{te} + (1-\omega)\,(\boldsymbol{1}_{n_s}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{1}_{n_s}^\top)^{-1}(\boldsymbol{1}_{n_s}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{y}^{tr}) \\ &= \omega\, b_{te} + (1-\omega)\widehat{\beta_0},\end{aligned} \tag{4.2.14}$$

the scalar analog of the convex combination (4.2.7) for $\omega = \lambda_\beta/[\lambda_Z \boldsymbol{1}_{n_s}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{1}_{n_s} + \lambda_\beta] \in (0, 1)$.

The effect of the β_0 prior precision, λ_β, on the β_0 posterior mean, $\mu_{\beta|tr}$, can be substantial. Suppose there is a *fixed process precision* λ_Z. As the prior precision increases, i.e., $\lambda_\beta \to \infty$, then $\omega \to 1$ and $\mu_{\beta|tr} \to b_\beta$. The posterior mean of β_0 guesses the prior mean and ignores the data. Similarly, as the β_0 prior precision decreases, i.e., $\lambda_\beta \to 0$, then $\omega \to 0$ and $\mu_{\beta|tr} \to \widehat{\beta_0}$ so that the predictor uses only the data and ignores the prior information. Calculation gives $\widehat{\beta_0} = 0.372$ for the training data in Fig. 4.1. Fix $b_\beta = 5$ and $\lambda_Z = 6$. Then $\mu_{\beta|tr} \to 5$ as $\lambda_\beta \to \infty$ and $\mu_{\beta|tr} \to 0.372$ as $\lambda_\beta \to 0$.

In contrast, the impact of the λ_β prior precision on the mean of the posterior of $Y(\boldsymbol{x}^{te})$, $\mu_{te|tr}(x^{te})$, can be relatively minor. Consider the same prior as in the previous paragraph and the training data of Fig. 4.1. Figure 4.1 shows that the effect of changing the prior precision on $\mu_{te|tr}(x^{te})$ is small. This behavior can be seen analytically from

$$\mu_{te|tr}(x^{te}) = \boldsymbol{r}_{te}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{y}^{tr} + (1 - \boldsymbol{r}_{te}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{1}_{n_s})\,\mu_{\beta|tr} \tag{4.2.15}$$

which shows that $\mu_{te|tr}(x^{te})$ depends on the prior only through the posterior mean $\mu_{\beta|tr}$ which is multiplied by the factor $(1 - \boldsymbol{r}_{te}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{1}_{n_s})$. Figure 4.2 shows that the factor $(1 - \boldsymbol{r}_{te}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{1}_{n_s})$ is very small in the center of the training data and even at its extremes only rises to about 0.04. Thus $\mu_{te|tr}(x^{te})$ depends primarily on the first term of (4.2.15) and hence is relatively unaffected by λ_β. ♦

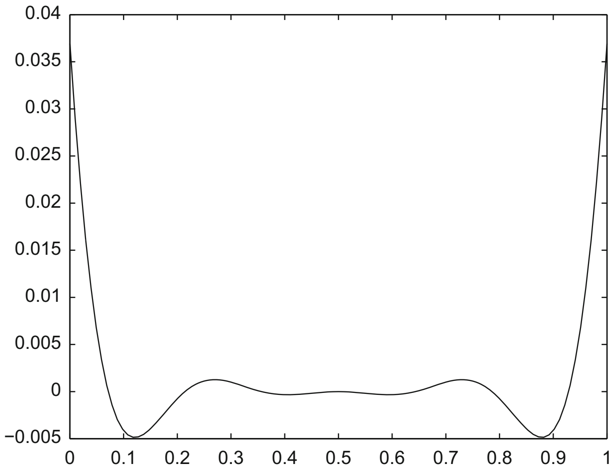


Fig. 4.2 The factor $(1 - \boldsymbol{r}_{te}^{\top}\boldsymbol{R}_{tr}^{-1}\boldsymbol{1}_{n_s})$ versus $x^{te} \in (0, 1)$

4.2.2 Posterior Inference for Model (4.1.1) When $\boldsymbol{\vartheta} = (\boldsymbol{\beta}, \lambda_Z)$

This section considers a more challenging model where both $\boldsymbol{\beta}$ and λ_Z in the regression + stationary GP model are *unknown*. Theorem 4.2 provides the predictive distribution of $\boldsymbol{Y}^{te}$ for informative and non-informative $(\boldsymbol{\beta}, \lambda_Z)$ priors. In both cases the $[\boldsymbol{Y}^{te} \mid \boldsymbol{Y}^{tr}]$ posterior distribution is a location shifted and scaled multivariate t distribution (see Appendix B.4 for the definition of this distribution). The degrees of freedom are greater when either $\boldsymbol{\beta}$ or λ_Z have an informative prior than when they have a non-informative one.

The informative prior considered in this subsection is stated in terms of the two factors of $[\boldsymbol{\beta}, \lambda_Z] = [\boldsymbol{\beta} \mid \lambda_Z]\times[\lambda_Z]$. Theorem 4.2 assumes that $[\boldsymbol{\beta} \mid \lambda_Z]$ is the multivariate normal distribution with known mean $\boldsymbol{b}_\beta$ and known scale matrix $\boldsymbol{V}_\beta$. Lacking more definitive information, $\boldsymbol{V}_\beta$ is often taken to be diagonal, if not simply the identity matrix. The marginal symmetry of each normal prior model makes strong assumptions; for example, it says that each component of $\boldsymbol{\beta}$ is equally likely to be less than or greater than the corresponding component of $\boldsymbol{b}_\beta$. The non-informative $\boldsymbol{\beta}$ prior is taken to be the (improper) intuitive choice

$$\pi(\boldsymbol{\beta} \mid \lambda_Z) \propto 1$$

used in Theorem 4.1.

The informative $[\lambda_Z]$ prior is taken to be a gamma distribution with specified mean and variance. The gamma prior can be made quite diffuse when its parameters yield a large mean and variance; operationally such a prior can be viewed as "non-informative." A more familiar non-informative prior for λ_Z is "Jeffreys prior"

$$\pi(\lambda_Z) \propto \frac{1}{\lambda_Z}, \quad \lambda_Z > 0$$

(see Jeffreys (1961), who gives arguments for this choice).

Theorem 4.2. Suppose $(\boldsymbol{Y}^{te}, \boldsymbol{Y}^{tr})$ follows the two-stage conditional model in which $[(\boldsymbol{Y}^{te}, \boldsymbol{Y}^{tr}) \mid (\boldsymbol{\beta}, \lambda_Z)]$ is given by (4.1.1), $n_s > p$, and all correlations are known.

(a) If $[\boldsymbol{\beta}, \lambda_Z]$ has prior specified by

$$[\boldsymbol{\beta} \mid \lambda_Z] \sim N_p\left(\boldsymbol{b}_\beta, \lambda_Z^{-1}\boldsymbol{V}_\beta\right) \quad \text{and} \quad [\lambda_Z] \sim \Gamma(c, d)$$

with known $\boldsymbol{b}_\beta$, $\boldsymbol{V}_\beta$, c, and d, then the posterior distributions of $\boldsymbol{\beta}$ and λ_Z are

$$[\lambda_Z \mid \boldsymbol{Y}^{tr} = \boldsymbol{y}^{tr}] \sim \Gamma((2c + n_s)/2, d_1^a) \text{ and}$$
$$[\boldsymbol{\beta} \mid \boldsymbol{Y}^{tr} = \boldsymbol{y}^{tr}] \sim T_p\left(2c + n_s,\ \boldsymbol{\mu}_{\beta|tr},\ 2d_1^a \boldsymbol{\Sigma}_{\beta|tr}/(2c + n_s)\right),$$

where

- $d_1^a = \left(2d + \left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right)^\top \boldsymbol{R}_{tr}^{-1}\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right) + \left(\widehat{\boldsymbol{\beta}} - \boldsymbol{b}_\beta\right)^\top \boldsymbol{\Sigma}_\pi^{-1}\left(\widehat{\boldsymbol{\beta}} - \boldsymbol{b}_\beta\right)\right)/2,$
- $\boldsymbol{G} = \boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr},$
- $\widehat{\boldsymbol{\beta}} = \boldsymbol{G}^{-1}\boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{y}^{tr},$
- $\boldsymbol{\Sigma}_\pi = \boldsymbol{G}^{-1} + \boldsymbol{V}_\beta,$
- $\boldsymbol{\Sigma}_{\beta|tr} = \left(\boldsymbol{G} + \boldsymbol{V}_\beta^{-1}\right)^{-1}$, and
- $\boldsymbol{\mu}_{\beta|tr} = \left(\boldsymbol{G} + \boldsymbol{V}_\beta^{-1}\right)^{-1}\left(\boldsymbol{G}\widehat{\boldsymbol{\beta}} + \boldsymbol{V}_\beta^{-1}\boldsymbol{b}_\beta\right) = \boldsymbol{\Sigma}_{\beta|tr}\left(\boldsymbol{G}\widehat{\boldsymbol{\beta}} + \boldsymbol{V}_\beta^{-1}\boldsymbol{b}_\beta\right).$

The predictive distribution of $\boldsymbol{Y}^{te}$ is

$$\left[\boldsymbol{Y}^{te} \,\middle|\, \boldsymbol{Y}^{tr} = \boldsymbol{y}^{tr}\right] \sim T_{n_e}\left(2c + n_s,\ \boldsymbol{\mu}_{te|tr},\ 2d_1^a \boldsymbol{M}_{te|tr}/(2c + n_s)\right)$$

where

- $\boldsymbol{\mu}_{te|tr} = \boldsymbol{F}_{te}\boldsymbol{\mu}_{\beta|tr} + \boldsymbol{R}_{te,tr}\boldsymbol{R}_{tr}^{-1}\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\boldsymbol{\mu}_{\beta|tr}\right),$
- $\boldsymbol{M}_{te|tr} = \boldsymbol{R}_{te} - \boldsymbol{R}_{te,tr}\boldsymbol{R}_{tr}^{-1}\boldsymbol{R}_{te,tr}^\top + \boldsymbol{H}_{te}\boldsymbol{\Sigma}_{\beta|tr}\boldsymbol{H}_{te}^\top$, and
- $\boldsymbol{H}_{te} = \boldsymbol{F}_{te} - \boldsymbol{R}_{te,tr}\boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr}.$

(b) Suppose the $[\boldsymbol{\beta}, \lambda_Z]$ prior is determined by independent $\boldsymbol{\beta}$ and λ_Z, where $[\boldsymbol{\beta}] \propto 1$, and $[\lambda_Z]$ has either the informative prior (b.1) or non-informative prior (b.2) as given in the following table:

$[\lambda_Z]$ prior	$\Gamma(c,d)$	$1/\lambda_Z$
Case designation	(b.1)	(b.2)

.

For (b.1) the posterior distributions of $\boldsymbol{\beta}$ and λ_Z are

$$[\lambda_Z \mid \boldsymbol{Y}^{tr} = \boldsymbol{y}^{tr}] \sim \Gamma((2c + n_s - p)/2, d_1^{b.1}) \text{ and}$$
$$[\boldsymbol{\beta} \mid \boldsymbol{Y}^{tr} = \boldsymbol{y}^{tr}] \sim T_p\left(2c + n_s - p,\ \widehat{\boldsymbol{\beta}},\ 2d_1^{b.1}\boldsymbol{G}^{-1}/(2c + n_s - p)\right)$$

where $\boldsymbol{G}$ and $\widehat{\boldsymbol{\beta}}$ are defined as for prior (a) and

$$d_1^{b.1} = \left(2d + \left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right)^\top \boldsymbol{R}_{tr}^{-1}\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right)\right)/2\,.$$

For (b.1) the predictive distribution of $\boldsymbol{Y}^{te}$ is

$$\left[\boldsymbol{Y}^{te} \mid \boldsymbol{Y}^{tr} = \boldsymbol{y}^{tr}\right] \sim T_{n_e}\left(2c + n_s - p,\ \boldsymbol{\mu}_{te|tr},\ 2d_1^{b.1}\boldsymbol{M}_{te|tr}/(2c + n_s - p)\right)$$

where

- $\boldsymbol{\mu}_{te|tr} = \boldsymbol{F}_{te}\widehat{\boldsymbol{\beta}} + \boldsymbol{R}_{te,tr}\boldsymbol{R}_{tr}^{-1}\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right)$, and
- $\boldsymbol{M}_{te|tr} = \boldsymbol{R}_{te} - \boldsymbol{R}_{te,tr}\boldsymbol{R}_{tr}^{-1}\boldsymbol{R}_{te,tr}^\top + \boldsymbol{H}_{te}\boldsymbol{G}^{-1}\boldsymbol{H}_{te}^\top$.

For (b.2) the posterior distributions of $\boldsymbol{\beta}$ and λ_Z are

$$[\lambda_Z \mid \boldsymbol{Y}^{tr} = \boldsymbol{y}^{tr}] \sim \Gamma((n_s - p)/2, d_1^{b.2}) \text{ and}$$
$$[\boldsymbol{\beta} \mid \boldsymbol{Y}^{tr} = \boldsymbol{y}^{tr}] \sim T_p\left(n_s - p,\ \widehat{\boldsymbol{\beta}},\ 2d_1^{b.2}\boldsymbol{G}^{-1}/(n_s - p)\right)$$

where $d_1^{b.2} = \left(\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right)^\top \boldsymbol{R}_{tr}^{-1}\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right)\right)/2$.
For (b.2) the predictive distribution of $\boldsymbol{Y}^{te}$ is

$$\left[\boldsymbol{Y}^{te} \mid \boldsymbol{Y}^{tr} = \boldsymbol{y}^{tr}\right] \sim T_{n_e}\left(n_s - p,\ \boldsymbol{\mu}_{te|tr},\ 2d_1^{b.2}\boldsymbol{M}_{te|tr}/(n_s - p)\right)$$

where $\boldsymbol{\mu}_{te|tr}$ and $\boldsymbol{M}_{te|tr}$ are defined as in (b.1).

At first glance, the formulas for the various posterior distributions given in Theorem 4.2 can leave the reader overwhelmed. However, there is intuition that can provide meaning to the formulas for the degrees of freedom, the mean, and the covariance matrix of the $\boldsymbol{Y}^{te}$ predictive distribution. Similar discernment can be provided concerning the $[\boldsymbol{\beta} \mid \boldsymbol{Y}^{tr}]$ and $[\lambda_Z \mid \boldsymbol{Y}^{tr}]$ posterior distributions. The insight is based on the fact that the mean and variance-covariance matrix of $\boldsymbol{W} \sim T_m(\nu, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ are $\boldsymbol{\mu}$ and $\nu\boldsymbol{\Sigma}/(\nu - 2)$, respectively, provided $\nu > 1$ and $\nu > 2$.

Starting with the *degrees of freedom* (dof) for the $\boldsymbol{Y}^{te}$ predictive distribution, regard its base value to be $n_s - p$ as would be the dof for an ordinary regression having p regressors and based on n_s observations. The base value is augmented by p additional degrees of freedom when $\boldsymbol{\beta}$ has the informative normal prior and is further augmented by $2c$ degrees of freedom when λ_Z has the informative gamma prior.

Thus the value of the dof is $n_s - p + p + 2c = 2c + n_s$ for prior (a) because both priors are informative. Similar formulas explain the two (b) prior cases. The bottom line is that inferences become "more precise" in the sense of larger values of the dof, when either of the priors is informative.

The *mean* of the $\boldsymbol{Y}^{te}$ predictive distribution is the same for the informative and non-informative $\boldsymbol{\beta}$ priors of Theorems 4.1 and 4.2. In the known λ_Z setting of Theorem 4.1, formulas (4.2.4) and (4.2.6) give the predictive mean to be

$$\boldsymbol{\mu}_{te|tr} = \boldsymbol{F}_{te}\,\boldsymbol{\mu}_{\beta|tr} + \boldsymbol{R}_{te,tr}\boldsymbol{R}_{tr}^{-1}\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\,\boldsymbol{\mu}_{\beta|tr}\right)$$

where $\boldsymbol{\mu}_{\beta|tr}$ is the mean of the conditional $[\boldsymbol{\beta} \mid \boldsymbol{Y}^{tr} = \boldsymbol{y}^{tr}]$ distribution, which depends on whether the $\boldsymbol{\beta}$ prior is informative or not. Examination of the $\boldsymbol{\mu}_{te|tr}$ formula for $\lambda_\beta = \lambda_Z$ shows it is identical to that of Theorem 4.2 for the two $\boldsymbol{\beta}$ prior cases. Thus the Bayesian predictor of the mean of $\boldsymbol{Y}^{te}$ is independent of the λ_Z prior for these two specific cases but depends on the $\boldsymbol{\beta}$ prior.

As in the discussion following Theorem 4.1, consider interpreting the posterior covariance when there is a *single test input* $\boldsymbol{x}^{te}$ which reduces the posterior covariance matrix to the real-valued posterior variance $\sigma^2_{te|tr}(\boldsymbol{x}^{te}) = Var(Y(\boldsymbol{x}^{te}) \mid \boldsymbol{Y}^{tr} = \boldsymbol{y}^{tr})$. Recall that when λ_Z is known and $\lambda_\beta = \lambda_Z$,

$$\sigma^2_{te|tr}(\boldsymbol{x}^{te}) = \lambda_Z^{-1}\left\{1 - \boldsymbol{r}_{te}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{r}_{te} + \boldsymbol{h}^\top \boldsymbol{Q}^{-1}\boldsymbol{h}\right\}, \tag{4.2.16}$$

where $\boldsymbol{h} = \boldsymbol{f}(\boldsymbol{x}^{te}) - \boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{r}_{te}$ and

$$\boldsymbol{Q} = \boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr} + \boldsymbol{V}_\beta^{-1} \quad \text{or} \quad \boldsymbol{Q} = \boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr},$$

according as the $\boldsymbol{\beta}$ prior is informative or non-informative. Now compare (4.2.16) to the posterior variance in Theorem 4.2 for unknown λ_Z. For simplicity the expressions below assume $[\boldsymbol{\beta}] \propto 1$, but a similar intuition holds for the informative $\boldsymbol{\beta}$ prior. For the gamma and Jeffreys prior, respectively, algebra shows that

$$\sigma^2_{te|tr}(\boldsymbol{x}^{te}) = K\left\{1 - \boldsymbol{r}_{te}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{r}_{te} + \boldsymbol{h}^\top\left(\boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr}\right)^{-1}\boldsymbol{h}\right\}, \tag{4.2.17}$$

where $K = \left(2d + \left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right)^\top \boldsymbol{R}_{tr}^{-1}\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right)\right)/(2c + n_s - p)$ (for the gamma prior) and $K = \left(\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right)^\top \boldsymbol{R}_{tr}^{-1}\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right)\right)/(n_s - p)$ (for Jeffreys prior). The factor of (4.2.17) in braces is the same as that in (4.2.16). Both K expressions can be viewed as estimates of λ_Z^{-1}, the $Y(\boldsymbol{x})$ process variance. For the Jeffreys prior, K is the weighted residual sum of squares

$$\frac{\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right)^\top \boldsymbol{R}_{tr}^{-1}\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right)}{n_s - p} \tag{4.2.18}$$

divided by the usual degrees of freedom and is recognizable as the frequentist estimate of λ_Z^{-1}. For the gamma prior,

$$K = \frac{d}{c} \times \frac{2c}{2c + n_s - p} + \frac{\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right)^\top \boldsymbol{R}_{tr}^{-1} \left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right)}{n_s - p} \times \frac{n_s - p}{2c + n_s - p}$$

which is a convex combination of the frequentist estimate (4.2.18) and d/c. The ratio c/d is the prior mean of λ_Z, and hence its reciprocal is a prior estimate of λ_Z^{-1}. For fixed c and d, the weight on the frequentist component increases as the amount of data, n_s, increases. When the ratio d/c is fixed, the weight on d/c increases as $c \to \infty$, which makes intuitive sense because the prior precision increases in this case.

Using the posterior variance formulas for the case of a single input $\boldsymbol{x}^{te}$, Theorem 4.2 gives the $100(1-\alpha)\%$ pointwise prediction bands,

$$P\left\{ Y(\boldsymbol{x}^{te}) \in \mu_{te|tr}(\boldsymbol{x}^{te}) \pm \sigma_{te|tr}(\boldsymbol{x}^{te})\, t_{\text{dof}}^{\alpha/2} \;\middle|\; \boldsymbol{Y}^{tr} \right\} = 1 - \alpha,$$

for $y(\boldsymbol{x}^{te})$ where $t_{\text{dof}}^{\alpha/2}$ is the upper $\alpha/2$ critical point of the standard t distribution with dof degrees of freedom (see Appendix A), and dof $= 2c + n_s - p$ or $n_s - p$ according as the informative or non-informative prior is appropriate for λ_Z.

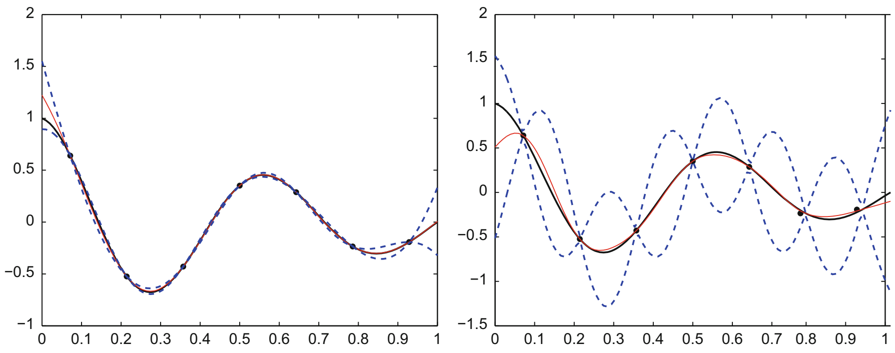


Fig. 4.3 The left and right panels plot, as base information, the damped cosine function and the $n_s = 7$ training data points (as solid circles) of Example 4.1. Both panels plot, as dashed blue curves, 95% pointwise prediction bands (4.2.19) for $y(x^{te})$ at $n_e = 103$ equally spaced x^{te} values in $(0, 1)$ but based on different assumed correlation structures. The left panel intervals assume $\xi = 10$ in (4.2.20), while the right panel intervals assume $\xi = 75$ in (4.2.20)

Example 4.1 (Continued). Recall the damped cosine function and training data of Example 4.1 which are replotted in both panels of Fig. 4.3. The left and right panels show the 95% pointwise prediction bands

$$\mu_{te|tr} \pm t_6^{0.05/2} \times \sqrt{\frac{\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right)^\top \boldsymbol{R}_{tr}^{-1} \left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right)}{7-1} \times \left\{1 - \boldsymbol{r}_{te}^\top \boldsymbol{R}_{tr}^{-1} \boldsymbol{r}_{te} + h^2/Q\right\}} \quad (4.2.19)$$

for the non-informative prior (b.2) of Theorem 4.2 but for different $Y(\boldsymbol{x})$ models. In both panels the first stage of the $Y(\boldsymbol{x})$ model assumes a GP with constant mean

($p = 1$) and correlation function of the (Gaussian) form

$$R(w \mid \xi) = \exp\left\{-\xi\, w^2\right\}. \tag{4.2.20}$$

The left panel bands were computed under the assumption that $\xi = 10.0$, and the right panel bands assumed $\xi = 75.0$. When $x_1 \neq x_2$ the $Y(x_1)$ and $Y(x_2)$ correlation is (much) nearer zero, i.e., nearer independence, for $\xi = 75.0$ (right panel) than for $\xi = 10.0$ (left panel). Intuitively, the prediction bands should be wider when the $Y(x)$ model is closer to the independence model than for the model which states that the $Y(x)$ march more closely in lockstep. This feature is clearly seen by comparing the left and right panels of Fig. 4.3.

Three other features can be seen in Fig. 4.3. For *any* ξ, the bands have *zero width* at each of the training data inputs. The predictor $\mu_{te|tr}(x^{te})$ is relatively insensitive to the choice of ξ for any x^{te} that *interpolates* the training data. For inputs that *extrapolate* the training data, here x^{te} near 0 or 1, the choice of ξ can substantially impact the value of $\mu_{te|tr}(x^{te})$. ♦

4.3 Inference for Non-conjugate Bayesian Models

This section describes Bayesian posterior inference for the (non-conjugate) regression + stationary GP model (4.1.1) of Sect. 4.1 in which $\boldsymbol{\vartheta} = (\boldsymbol{\beta}, \lambda_Z, \boldsymbol{\kappa})$ are unknown parameters. Sections 4.2.1 and 4.2.2 assumed that the correlation function, $R(\cdot \mid \boldsymbol{\kappa})$, of the stationary interpolating GP was known and hence the correlation matrices $\boldsymbol{R}_{te}$, $\boldsymbol{R}_{tr}$, and $\boldsymbol{R}_{te,tr}$ in (4.1.1) were also known. This section drops the assumption that $\boldsymbol{\kappa}$ is known.

For the remainder of these introductory paragraphs, assume it is desired to predict $y(\boldsymbol{x})$ at the *single input* $\boldsymbol{x}^{te}$. To illustrate the difficulties of the unknown $\boldsymbol{\kappa}$ case, recall that when $\boldsymbol{\kappa}$ is *known*, Theorems 4.1 and 4.2 give, for certain priors, expressions for the Bayesian predictor of $y(\boldsymbol{x}^{te})$, $\mu_{te|tr}(\boldsymbol{x}^{te}) = \mu_{te|tr}(\boldsymbol{x}^{te} \mid \boldsymbol{\kappa})$, and for the posterior variance (MSPE) of the predictive distribution,

$$\sigma^2_{te|tr}(\boldsymbol{x}^{te}) = \sigma^2_{te|tr}(\boldsymbol{x}^{te} \mid \boldsymbol{\kappa}) = E\left[\left(\mu_{te|tr}(\boldsymbol{x}^{te} \mid \boldsymbol{\kappa}) - Y(\boldsymbol{x}^{te})\right)^2 \,\middle|\, \boldsymbol{Y}^{tr}\right],$$

which is a measure of the model uncertainty in the prediction. Both expressions $\mu_{te|tr}(\boldsymbol{x}^{te} \mid \boldsymbol{\kappa})$ and $\sigma^2_{te|tr}(\boldsymbol{x}^{te} \mid \boldsymbol{\kappa})$ explicitly show that $\boldsymbol{\kappa}$ is known by the conditioning notation.

When $\boldsymbol{\kappa}$ is *unknown* and estimated by $\widehat{\boldsymbol{\kappa}}$, a "natural" predictor of $y(\boldsymbol{x}^{te})$ is the plug-in expression $\mu_{te|tr}(\boldsymbol{x}^{te} \mid \widehat{\boldsymbol{\kappa}})$. A naive quantification of the uncertainty in $\mu_{te|tr}(\boldsymbol{x}^{te} \mid \widehat{\boldsymbol{\kappa}})$ is the plug-in variance, $\sigma^2_{te|tr}(\boldsymbol{x}^{te} \mid \widehat{\boldsymbol{\kappa}})$. However, the plug-in posterior variance is different than

$$\text{MSPE}(\mu_{te|tr}(\boldsymbol{x}^{te} \mid \widehat{\boldsymbol{\kappa}}), \boldsymbol{\kappa}) = E_{\boldsymbol{\kappa}}\left[\left(\mu_{te|tr}(\boldsymbol{x}^{te} \mid \widehat{\boldsymbol{\kappa}}) - Y(\boldsymbol{x}^{te})\right)^2\right],$$

which is the frequentist expression for the uncertainty of $\mu_{te|tr}(\boldsymbol{x}^{te} \,|\, \widehat{\boldsymbol{\kappa}})$. Zimmerman and Cressie (1992) show that when the underlying $Y(\boldsymbol{x})$ is a GP,

$$\sigma^2_{te|tr}(\boldsymbol{x}^{te} \,|\, \widehat{\boldsymbol{\kappa}}) \le \mathrm{MSPE}(\mu_{te|tr}(\boldsymbol{x}^{te} \,|\, \widehat{\boldsymbol{\kappa}}), \boldsymbol{\kappa}) \qquad (4.3.1)$$

under mild conditions. Thus the naive estimator $\sigma^2_{te|tr}(\boldsymbol{x}^{te} \,|\, \widehat{\boldsymbol{\kappa}})$ *underestimates* the true uncertainty of the plug-in predictor. The amount of the underestimate is most severe when the underlying GP has weak correlation.

This section describes the Bayesian alternative to plug-in methodology for analyzing the case of unknown $\boldsymbol{\kappa}$. The Bayesian approach assumes that a prior distribution is available for $\boldsymbol{\kappa}$ that provides information about likely values of this parameter. As motivation for the practical benefit of this approach, Handcock and Stein (1993) analytically carried out the Bayesian calculations for a specific two-input example using several regression models and correlation functions. They reported that, as suggested by (4.3.1), for most cases studied the $y(\boldsymbol{x}^{te})$ prediction bands based on the Bayesian predictor and Bayesian variance were wider than the intervals based on plug-in quantities. The plug-in interval had particularly poor performance relative to the Bayes interval when $\widehat{\boldsymbol{\kappa}}$ was determined by an eye-fit to the "variogram" associated with the correlation function.

4.3.1 The Hierarchical Bayesian Model and Posterior

The first stage of the assumed hierarchical Bayesian model is the (conditional) regression + stationary GP model of Sect. 4.1 which assumes that the training and test data be modeled as a draw from

$$\left[\begin{pmatrix} \boldsymbol{Y}^{te} \\ \boldsymbol{Y}^{tr} \end{pmatrix} \middle| \, \boldsymbol{\vartheta} \right] \sim N_{n_e+n_s}\left(\begin{pmatrix} \boldsymbol{F}_{te} \\ \boldsymbol{F}_{tr} \end{pmatrix} \boldsymbol{\beta} ,\ \lambda_Z^{-1} \begin{bmatrix} \boldsymbol{R}_{te} & \boldsymbol{R}_{te,tr} \\ \boldsymbol{R}_{te,tr}^\top & \boldsymbol{R}_{tr} \end{bmatrix} \right)$$

where $\boldsymbol{\vartheta} \equiv (\boldsymbol{\beta}, \lambda_Z, \boldsymbol{\kappa})$.

The second stage of the hierarchical Bayesian model specifies the $[\boldsymbol{\vartheta}]$ prior. Below, p denotes the number of regression coefficients and d the number of inputs. This chapter assumes that knowledge about the regressor $\boldsymbol{\beta}$ and scale parameter λ_Z are independent of the correlation information $\boldsymbol{\kappa}$ so that

$$[\boldsymbol{\vartheta}] = [\boldsymbol{\beta}, \lambda_Z, \boldsymbol{\kappa}] = [\boldsymbol{\beta}, \lambda_Z][\boldsymbol{\kappa}]. \qquad (4.3.2)$$

Example 4.2 (An Informative Prior). Suppose that the correlations among the components of $Y(\boldsymbol{x})$ are specified by the separable Gaussian correlation function

$$R(\boldsymbol{h}) = \prod_{k=1}^{d} \rho_k^{h_k^2} \qquad (4.3.3)$$

so that $\boldsymbol{\kappa} = (\rho_1, \ldots, \rho_d)$. Recall that (4.3.3) uses one of the equivalent versions of the Gaussian correlation function given in (2.2.9). Consider the informative prior in Part (a) of Theorem 4.2 which uses $[\boldsymbol{\beta}, \lambda_Z] = [\boldsymbol{\beta} \mid \lambda_Z]\,[\lambda_Z]$ and specifies

$$[\boldsymbol{\beta} \mid \lambda_Z] \sim N_p\left(\boldsymbol{b}_\beta, \lambda_Z^{-1}\boldsymbol{V}_\beta\right) \quad \text{and} \quad [\lambda_Z] \sim \Gamma(c, d) \tag{4.3.4}$$

with known $\boldsymbol{b}_\beta$, $\boldsymbol{V}_\beta$, and (c, d). Additionally, suppose the $(\rho_1, \ldots, \rho_d)$ prior chooses the d components independently and identically distributed as $Be(a_\rho, b_\rho)$ with specified $a_\rho > 0$ and $b_\rho > 0$ (hence with common known mean and variance). Then under the assumption (4.3.2), the $[\boldsymbol{\beta}, \lambda_Z, \boldsymbol{\kappa}]$ prior density is proportional to

$$\begin{aligned} &\lambda_Z^{p/2} \exp\left(-\frac{\lambda_Z}{2}(\boldsymbol{\beta} - \boldsymbol{b}_\beta)^\top \boldsymbol{V}_\beta^{-1}(\boldsymbol{\beta} - \boldsymbol{b}_\beta)\right) \times \lambda_Z^{c-1} \exp(-d\lambda_Z) \\ &\quad \times \prod_{k=1}^{d} \rho_k^{a_\rho - 1}(1 - \rho_k)^{b_\rho - 1} . \end{aligned} \tag{4.3.5}$$

If subject matter considerations suggest *different* prior means and uncertainties for the correlation parameters, the prior (4.3.5) can modified in a straightforward manner as long as *independent* beta distributions provide a reasonable description of the individual correlations. Of course more complicated $[\rho_1, \ldots, \rho_d]$ priors may also be required in some applications. ♦

Example 4.3 (A Non-informative Prior). Assume the Gaussian correlation function (4.3.3) of Example 4.2 and the non-informative $[\boldsymbol{\beta}, \lambda_Z]$ prior

$$[\boldsymbol{\beta}, \lambda_Z] \propto \frac{1}{\lambda_Z}$$

from Part (b) of Theorem 4.2. Using the same beta prior for $(\rho_1, \ldots, \rho_d)$ as in (4.3.5), the $[\boldsymbol{\beta}, \lambda_Z, \boldsymbol{\kappa}]$ prior density is proportional to

$$\frac{1}{\lambda_Z} \times \prod_{k=1}^{d} \rho_k^{a_\rho - 1}(1 - \rho_k)^{b_\rho - 1} .$$

In this example and in Example 4.2, it is useful to regard the t posterior distributions given in Theorem 4.2 as conditional on $\boldsymbol{\kappa}$ and indicate this fact by the notation $[\boldsymbol{Y}^{te} \mid \boldsymbol{Y}^{tr}, \boldsymbol{\kappa}]$. ♦

The information about $\boldsymbol{Y}^{te}$ contained in $\boldsymbol{Y}^{tr}$ is specified by the predictive distribution

$$\begin{aligned} \left[\boldsymbol{Y}^{te} \mid \boldsymbol{Y}^{tr}\right] &= \int \left[\boldsymbol{Y}^{te}, (\boldsymbol{\beta}, \lambda_Z, \boldsymbol{\kappa}) \mid \boldsymbol{Y}^{tr}\right] d\boldsymbol{\beta}\, d\lambda_Z\, d\boldsymbol{\kappa} \\ &= \int \left(\int \left[\boldsymbol{Y}^{te} \mid \boldsymbol{Y}^{tr}, (\boldsymbol{\beta}, \lambda_Z, \boldsymbol{\kappa})\right] \left[(\boldsymbol{\beta}, \lambda_Z, \boldsymbol{\kappa}) \mid \boldsymbol{Y}^{tr}\right] d\boldsymbol{\beta}\, d\lambda_Z \right) d\boldsymbol{\kappa} . \end{aligned} \tag{4.3.6}$$

As noted in Examples 4.2 and 4.3, the integral (4.3.6) can be simplified in cases where the $[\boldsymbol{\beta}, \lambda_Z]$ prior satisfies (one of) the assumptions in Theorem 4.2 because the inner integral has a closed form. The predictive distribution becomes

$$\left[\boldsymbol{Y}^{te} \mid \boldsymbol{Y}^{tr}\right] = \int \left[\boldsymbol{Y}^{te}, \boldsymbol{\kappa} \mid \boldsymbol{Y}^{tr}\right] d\boldsymbol{\kappa} = \int \left[\boldsymbol{Y}^{te} \mid \boldsymbol{Y}^{tr}, \boldsymbol{\kappa}\right]\left[\boldsymbol{\kappa} \mid \boldsymbol{Y}^{tr}\right] d\boldsymbol{\kappa} \tag{4.3.7}$$

where $[\boldsymbol{Y}^{te} \mid \boldsymbol{Y}^{tr}, \boldsymbol{\kappa}]$ is the appropriate t density from Theorem 4.2. However, even the simpler integration (4.3.7) can be prohibitive, usually because of the intractability of the integrand or the complicated $\boldsymbol{\kappa}$ prior.

Bayesian inference about the model parameters is based on the posterior of the parameters. For example, the marginal posterior of the correlation parameters can be computed using

$$\left[\boldsymbol{\kappa} \mid \boldsymbol{Y}^{tr}\right] = \int \left[\boldsymbol{\beta}, \lambda_Z, \boldsymbol{\kappa} \mid \boldsymbol{Y}^{tr}\right] d\boldsymbol{\beta}\, d\lambda_Z, \tag{4.3.8}$$

where the integrand in (4.3.8) is determined from

$$\left[\boldsymbol{\beta}, \lambda_Z, \boldsymbol{\kappa} \mid \boldsymbol{Y}^{tr}\right] \propto \left[\boldsymbol{Y}^{tr} \mid \boldsymbol{\beta}, \lambda_Z, \boldsymbol{\kappa}\right]\left[\boldsymbol{\beta}, \lambda_Z, \boldsymbol{\kappa}\right].$$

The marginal posterior of $\boldsymbol{\beta}$ or λ_Z or the joint posterior of combinations of these parameters can be obtained in a similar fashion. In particular, Eq. (4.3.8) for the posterior of $\boldsymbol{\kappa}$ involves a $p+1$ dimensional integration which is ordinarily less complicated than the integration (4.3.6) and can be carried out in closed form for "simple" priors.

Example 4.2 (Continued). Consider an *arbitrary prior*, $[\boldsymbol{\kappa}]$, for the $(\rho_1, \ldots, \rho_d)$ correlations in (4.3.3), and assume the informative prior specified by (4.3.4) for $[\boldsymbol{\beta}, \lambda_Z]$. It can be shown that the (marginal) $[\boldsymbol{\kappa} \mid \boldsymbol{Y}^{tr}]$ posterior has kernel

$$\left[\boldsymbol{\kappa} \mid \boldsymbol{Y}^{tr}\right] \propto \frac{[\boldsymbol{\kappa}]}{(b_1)^{a_1}\, (\det(\boldsymbol{R}_{tr}))^{1/2}\, (\det(\boldsymbol{G}))^{1/2}\, (\det(\boldsymbol{\Sigma}_\pi))^{1/2}} \tag{4.3.9}$$

where

- $a_1 = (2c + n_s)/2$,
- $b_1 = \left(2d + (\boldsymbol{Y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}})^\top \boldsymbol{R}_{tr}^{-1}(\boldsymbol{Y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}) + (\widehat{\boldsymbol{\beta}} - \boldsymbol{b}_\beta)^\top \boldsymbol{\Sigma}_\pi^{-1}(\widehat{\boldsymbol{\beta}} - \boldsymbol{b}_\beta)\right)/2 = d_1^a$,
- $\boldsymbol{G} = \boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1} \boldsymbol{F}_{tr}$
- $\widehat{\boldsymbol{\beta}} = \boldsymbol{G}^{-1}\boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{Y}^{tr}$, and
- $\boldsymbol{\Sigma}_\pi = \boldsymbol{G}^{-1} + \boldsymbol{V}_\beta$.

When a non-informative prior in $\boldsymbol{\beta}$ is specified, $[\boldsymbol{\beta}] \propto 1$, an analogous expression for (4.3.9) can be derived by setting $\det(\boldsymbol{\Sigma}_\pi) = 1$. For (b.1) in Theorem 4.2, $a_1 = (2c+n_s-p)/2$ and $b_1 = d_1^{b.1}$. For (b.2) in Theorem 4.2, $a_1 = (n_s-p)/2$ and $b_1 = d_1^{b.2}$. The ρ parameters in $[\boldsymbol{\kappa} \mid \boldsymbol{Y}^{tr}]$ appear in $\boldsymbol{R}_{tr}$, and so (4.3.9) is analytically intractable for most $[\boldsymbol{\kappa}]$ prior distributions. Markov chain Monte Carlo (MCMC) algorithms, two of which are described in Appendix D, give one option to sample $[\boldsymbol{\kappa} \mid \boldsymbol{Y}^{tr}]$ numerically. The mean and standard deviation of the $\boldsymbol{\kappa}$ posterior distribution give a Bayesian estimate of $\boldsymbol{\kappa}$ and a quantification of the uncertainty in estimating $\boldsymbol{\kappa}$. ♦

As suggested by (4.3.9), there does not exist a familiar closed form density for $[\boldsymbol{\beta}, \lambda_Z, \boldsymbol{\kappa} \mid \boldsymbol{Y}^{tr}]$. In practice, a combination of two MCMC algorithms can be employed to create draws from the $[\boldsymbol{\beta}, \lambda_Z, \boldsymbol{\kappa} \mid \boldsymbol{Y}^{tr}]$ posterior distribution: the Gibbs algorithm and the Metropolis–Hastings (MH) algorithm (Appendix D). These draws, for example, can be used to estimate the marginal $[\boldsymbol{\kappa} \mid \boldsymbol{Y}^{tr}]$ posterior. Additionally the (possibly high-dimensional) integrals (4.3.6) (or 4.3.7) can be approximated to determine $[\boldsymbol{Y}^{te} \mid \boldsymbol{Y}^{tr}]$. Once $[\boldsymbol{Y}^{te} \mid \boldsymbol{Y}^{tr}]$ is approximately determined, the Bayesian alternatives to $\boldsymbol{\mu}_{te|tr}(\widehat{\boldsymbol{\kappa}})$ and $\boldsymbol{\Sigma}_{te|tr}(\widehat{\boldsymbol{\kappa}})$,

$$E\left[\boldsymbol{Y}^{te} \mid \boldsymbol{Y}^{tr}\right] \quad \text{and}$$

$$\boldsymbol{\Sigma}_{te|tr} \equiv E\left[\left(\boldsymbol{Y}^{te} - E\left[\boldsymbol{Y}^{te} \mid \boldsymbol{Y}^{tr}\right]\right)\left(\boldsymbol{Y}^{te} - E\left[\boldsymbol{Y}^{te} \mid \boldsymbol{Y}^{tr}\right]\right)^{\top} \mid \boldsymbol{Y}^{tr}\right],$$

can be estimated.

4.3.2 Predicting Failure Depths of Sheet Metal Pockets

Example 1.2 (Continued). Recall that Montgomery and Truss (2001) described a simulator that computes the failure depth of a symmetric rectangular pocket that is punched in automobile steel sheets as a function of $d = 6$ manufacturing conditions. This example compares the EBLUP and (fully) Bayesian predictors based on a 60 run set of training data drawn from the 234 simulator runs that are available. The remaining 174 simulator runs are used as test data to compare the predictions of the two predictors and the corresponding coverages.

The 60 run training data was selected by standardizing all inputs to $[0, 1]$ and choosing that subset which approximately maximized the minimum Euclidean interpoint distance among all $\binom{234}{60}$ subsets of size 60. Below, $\boldsymbol{y}^{60}$ denotes the 60 run training data set and $\boldsymbol{y}^{174}$ the true values of the test data; the $\boldsymbol{y}^{174}$ values are used to compare prediction methodologies and their coverages.

The EBLUP predictions for this example are based on estimating the parameters in the model

$$[Y(\boldsymbol{x}) \mid \beta_0, \lambda_Z, \boldsymbol{\rho}] = \beta_0 + Z(\boldsymbol{x}) \tag{4.3.10}$$

where $Z(\boldsymbol{x})$ is a zero mean stationary GP with process precision λ_Z (in the notation of this section) and Gaussian correlation function

$$Cor\left[Y(\boldsymbol{x}_1), Y(\boldsymbol{x}_2)\right] = \prod_{\ell=1}^{6} \rho_\ell^{4\,(x_{1,\ell} - x_{2,\ell})^2}.$$

Scaling and shifting the $d = 6$ model inputs to [0,1] to provide numerical stability, the REML estimates of the ρ parameters are

$$(0.88, 0.97, 0.79, 0.87, 0.96, 0.99) \tag{4.3.11}$$

where the estimated ρ in (4.3.11) are in the order

1	Clearance (c)
2	Fillet radius (r)
3	Punch plan view radius (p)
4	Width of pocket (w)
5	Length of pocket (l)
6	Lock bead distance (d)

.

Formula (3.3.8) is used to estimate $\sigma_Z^2 \equiv 1/\lambda_Z$ as $\widehat{\sigma_Z^2} = 9263.2$, and the expression following (3.3.2) is used to estimate β_0 yielding $\widehat{\beta_0} = 22.8$. Figure 4.4 plots the simulated $\boldsymbol{y}^{174}$ values versus their predicted EBLUPs. Visually the predictions are symmetrically distributed about the line $y = x$ with similar magnitudes for the prediction errors of both large and small simulated failure depths. Quantitatively, the empirical root mean squared prediction error, the ERMSPE, of the REML-EBLUP is 17.46 (see definition (3.4.2)).

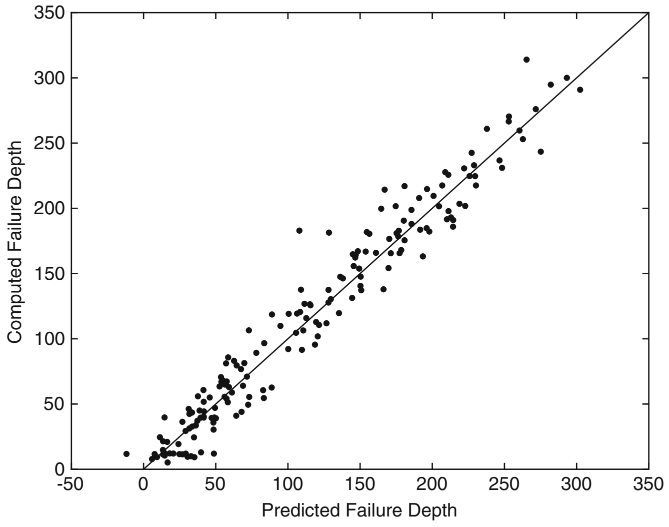


Fig. 4.4 Scatterplot of the 174 computed failure depths $\boldsymbol{y}^{174}$ versus their REML-EBLUP predictions; the line $y = x$

Letting $y(\boldsymbol{x}_1), \ldots, y(\boldsymbol{x}_{174})$ denote the 174 simulated test values, the empirical coverage of these test values by nominal pointwise 95% intervals is the proportion of the 174 simulated outputs that are covered by a 95% prediction interval, i.e.,

$$\frac{1}{174}\sum_{k=1}^{174} I\left\{ y(\boldsymbol{x}_k) \in \left(\widehat{y}(\boldsymbol{x}_k) \pm 2 \times s(\boldsymbol{x}_k)\right)\right\}$$

where $I\{\cdot\}$ is the indicator function which takes the value 1 if its argument is true and 0 otherwise. *For this test set, the coverage of the REML-EBLUP is* 61% *which is substantially less than the target value of* 95%.

Now consider fully Bayesian prediction. To facilitate the specification of the prior, the $\boldsymbol{y}^{60}$ training data are centered about their mean and scaled by their standard deviation yielding

$$e(\boldsymbol{x}_i) \equiv \frac{y(\boldsymbol{x}_i) - \text{Mean}(\boldsymbol{y}^{60})}{\text{StdDev}(\boldsymbol{y}^{60})}, \quad i = 1, \ldots, 60,$$

where Mean($\boldsymbol{y}^{60}$) and StdDev($\boldsymbol{y}^{60}$) are the sample mean and sample standard deviation of the training data, respectively. In terms of these (standardized) residuals, the $\boldsymbol{y}^{60}$ training data are

$$y(\boldsymbol{x}_i) = \text{Mean}(\boldsymbol{y}^{60}) + \text{StdDev}(\boldsymbol{y}^{60}) \times e(\boldsymbol{x}_i), \quad i = 1, \ldots, 60.$$

The vector of residuals

$$\boldsymbol{e}^{60} = (e(\boldsymbol{x}_1), \ldots, e(\boldsymbol{x}_{60}))^\top$$

has sample mean zero and sample standard deviation (and sample variance) equal to 1. Predictions will be made of the residuals for the 174-point test data.

The Bayesian model that is fit below regards $\boldsymbol{e}^{60}$ as a draw from $Y(\boldsymbol{x})$ where the likelihood stage of the model is

$$[Y(\boldsymbol{x}) \mid \lambda_Z, \boldsymbol{\rho}] = 0 + Z(\boldsymbol{x})$$

and $Z(\boldsymbol{x})$ is described below (4.3.10). To simplify sampling, the prior assumes independent λ_Z and $\boldsymbol{\rho}$ priors and, further, that the ρ_i, $i = 1, \ldots, 6$, are also independent. The λ_Z prior is selected to reflect the fact that values near 1 are consistent with the (sample) variance of $\boldsymbol{e}^{60}$. The ρ_i priors are assumed to be i.i.d. $Be(1, 0.1)$ distributions which therefore have mean $0.9 = 1/1.1$ but support over the entire interval (0, 1); the large mean values states that there is relatively little prior belief that each input is active. Putting the components together, the $[\lambda_Z, \boldsymbol{\rho}]$ prior used in this example is assumed *proportional to*

$$\lambda_Z^4 \exp(-5\lambda_Z) \times I_{[0.3,+\infty)}\{\lambda_Z\} \times \prod_{k=1}^{6} \frac{1}{(1-\rho_k)^{.9}},$$

where the indicator function $I_{[0.3,+\infty)}\{\lambda_Z\}$ truncates λ_Z to $(0.3, +\infty)$ to prevent posterior draws having extremely large $1/\lambda_Z$.

After applying Graves (2011) to tune the proposal distribution of the MH algorithm used to make draws from the $(\lambda_Z, \boldsymbol{\rho})$ posterior, this Bayesian model was fit using 10,500 burn-in draws followed by 10,000 production draws from the $(\lambda_Z, \boldsymbol{\rho})$ posterior (see Appendix D). The predictors and prediction sets are based on 200 equally spaced draws from the 10,000 posterior draws yielding $(\lambda_Z^s, \boldsymbol{\rho}^s)$, $s = 1, \ldots, 200$.

For each test data input $\boldsymbol{x}_i^{te}$, $i = 1, \ldots, 174$, and posterior parameter draw, the posterior mean

$$e^s(\boldsymbol{x}_i^{te}) = E\left[\, Y(\boldsymbol{x}_i^{te}) \;\middle|\; \lambda_Z^s, \boldsymbol{\rho}^s \right] = \boldsymbol{r}_s^\top \left(\boldsymbol{x}_i^{te}\right) \boldsymbol{R}_s^{-1} \boldsymbol{e}^{60}$$

was computed. The correlation parameter used to compute $\boldsymbol{r}_s$ and $\boldsymbol{R}_s$ is $\boldsymbol{\rho}^s$. The mean (or median) of $e^1(\boldsymbol{x}_i^{te}), \ldots, e^{200}(\boldsymbol{x}_i^{te})$ is the predicted value of $e(\boldsymbol{x}_i^{te})$, which is denoted $\widehat{\boldsymbol{e}}(\boldsymbol{x}_i^{te})$.

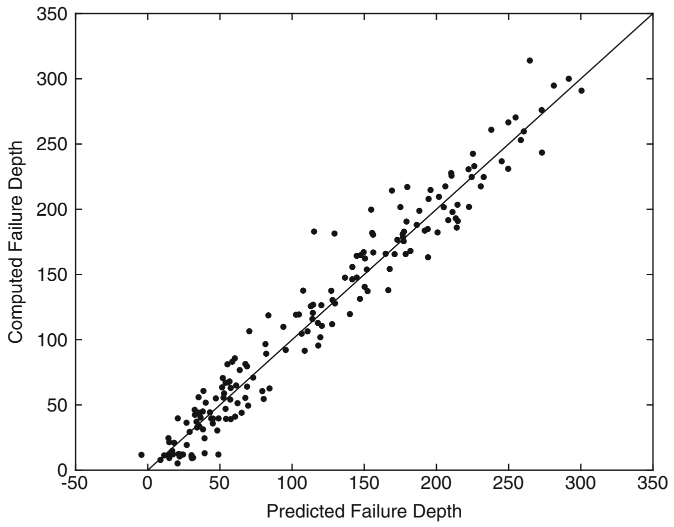


Fig. 4.5 Scatterplot of the 174 computed failure depths $\boldsymbol{y}^{174}$ versus their Bayesian predictions; the line $y = x$

Letting $\widehat{\boldsymbol{e}}^{174}$ denote the vector of predicted residuals for the 174 test inputs, the Bayesian prediction of $\boldsymbol{y}^{174}$ is

$$\widehat{\boldsymbol{y}}^{174} = \text{Mean}(\boldsymbol{y}^{60}) + \text{StdDev}(\boldsymbol{y}^{60}) \times \widehat{\boldsymbol{e}}^{174} \,. \tag{4.3.12}$$

Figure 4.5 plots the computed $\boldsymbol{y}^{174}$ versus their Bayesian predictions $\widehat{\boldsymbol{y}}^{174}$. The visual impression is similar to that for the REML-EBLUP predictions. The empirical root mean squared prediction error (ERMSPE) of the Bayesian predictor is 17.14 (which is virtually identical to the 17.46 ERMSPE for the REML-EBLUP predictor).

The strength of Bayesian predictors is that, compared with plug-in predictors which treat the correlation parameters as known, the uncertainty that the Bayesian methodology builds into its assessment tends to make coverage statements for predicted $y(\boldsymbol{x}_i^{te})$ based on wider intervals than those in Chap. 3. In this example the 2.5 and 97.5% sample quantiles of $e^1(\boldsymbol{x}_i^{te}), \ldots, e^{200}(\boldsymbol{x}_i^{te})$ form the basis of nominal 95% intervals for $y(\boldsymbol{x}_i^{te})$ based on (4.3.12). *In this case, calculation shows* 75% *of the* 174 *simulated values are contained in the resulting intervals.* While still less than the nominal 95%, the greater width of the Bayesian interval more correctly reflects the uncertainty in the correlation and precision parameters than the coverage provided by the EBLUP intervals. ♦

4.4 Chapter Notes

4.4.1 Outline of the Proofs of Theorems 4.1 and 4.2

The strategy for proving Theorems 4.1 and 4.2 was introduced at the beginning of Sect. 4.2. To review, suppose that $(\boldsymbol{Y}^{te}, \boldsymbol{Y}^{tr})$ has a conditional distribution $[(\boldsymbol{Y}^{te}, \boldsymbol{Y}^{tr}) \mid \boldsymbol{\vartheta}]$ given an unknown parameter $\boldsymbol{\vartheta}$ and the conditional distribution $[\boldsymbol{\vartheta} \mid \boldsymbol{Y}^{tr}]$ can be determined. (Throughout the notation suppresses any dependence on *known parameters*.) Then

$$\left[\boldsymbol{Y}^{te}, \boldsymbol{\vartheta} \mid \boldsymbol{Y}^{tr}\right] = \left[\boldsymbol{Y}^{te} \mid \boldsymbol{Y}^{tr}, \boldsymbol{\vartheta}\right] \times \left[\boldsymbol{\vartheta} \mid \boldsymbol{Y}^{tr}\right]$$

and calculating the $\boldsymbol{Y}^{te}$ marginal of $[\boldsymbol{Y}^{te}, \boldsymbol{\vartheta} \mid \boldsymbol{Y}^{tr}]$ provides the required predictive distribution.

Proof of Theorem 4.1

The predictive densities (4.2.3) and (4.2.6) in Theorem 4.1 require straightforward calculations in order to implement the right-hand integral of (4.1.2) which, in this case, is

$$\pi\left(\boldsymbol{y}^{te} \mid \boldsymbol{y}^{tr}\right) = \int \pi\left(\boldsymbol{y}^{te} \mid \boldsymbol{\beta}, \boldsymbol{y}^{tr}\right) \pi\left(\boldsymbol{\beta} \mid \boldsymbol{y}^{tr}\right) \, d\boldsymbol{\beta}. \tag{4.4.1}$$

During the process of calculating the integrand in (4.4.1), the appropriate posterior density $\pi(\boldsymbol{\beta} \mid \boldsymbol{y}^{tr})$ will be identified for the $\boldsymbol{\beta}$ priors considered by the theorem.

The left-hand factor in the integrand of (4.4.1) is the density $\pi(\boldsymbol{y}^{te} \mid \boldsymbol{\beta}, \boldsymbol{y}^{tr})$ which is immediate as the conditional normal distribution

$$\left[\boldsymbol{Y}^{te} \mid \boldsymbol{\beta}, \boldsymbol{y}^{tr}\right] \sim N_{n_e}\left(\boldsymbol{\mu}_{te|tr,\beta}, \boldsymbol{\Sigma}_{te|tr,\beta}\right), \tag{4.4.2}$$

where

$$\boldsymbol{\mu}_{te|tr,\beta} = \boldsymbol{F}_{te}\boldsymbol{\beta} + \boldsymbol{R}_{te,tr}\boldsymbol{R}_{tr}^{-1}(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\boldsymbol{\beta}) \text{ and } \boldsymbol{\Sigma}_{te|tr,\beta} = \lambda_Z^{-1}\left(\boldsymbol{R}_{te} - \boldsymbol{R}_{te,tr}\boldsymbol{R}_{tr}^{-1}\boldsymbol{R}_{te,tr}^{\top}\right)$$

(see Appendix B).

The right-hand factor $\pi(\boldsymbol{\beta} \mid \boldsymbol{y}^{tr})$ is derived by observing

$$\pi\left(\boldsymbol{\beta} \mid \boldsymbol{y}^{tr}\right) \propto \pi\left(\boldsymbol{y}^{tr} \mid \boldsymbol{\beta}\right) \times \pi\left(\boldsymbol{\beta}\right),$$

calculating the right-hand side, and retaining the terms involving $\boldsymbol{\beta}$ to determine the kernel of $\pi(\boldsymbol{\beta} \mid \boldsymbol{y}^{tr})$. For Case (a) this calculation yields

$$\begin{aligned}\pi\left(\boldsymbol{\beta} \mid \boldsymbol{y}^{tr}\right) &\propto \exp\left\{-\frac{1}{2}(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\boldsymbol{\beta})^\top \lambda_Z \boldsymbol{R}_{tr}^{-1}(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\boldsymbol{\beta}) \right.\\ &\qquad \left. -\frac{1}{2}(\boldsymbol{\beta} - \boldsymbol{b}_\beta)^\top \lambda_\beta \boldsymbol{V}_\beta^{-1}(\boldsymbol{\beta} - \boldsymbol{b}_\beta)\right\}\\ &\propto \exp\left\{-\frac{1}{2}\boldsymbol{\beta}^\top \boldsymbol{A}^{-1}\boldsymbol{\beta} + \boldsymbol{\nu}^\top\boldsymbol{\beta}\right\},\end{aligned}$$

where $\boldsymbol{A}^{-1} = \left[\lambda_Z \boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr} + \lambda_\beta \boldsymbol{V}_\beta^{-1}\right]$ and $\boldsymbol{\nu} = \left[\lambda_Z \boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{y}^{tr} + \lambda_\beta \boldsymbol{V}_\beta^{-1}\, \boldsymbol{b}_\beta\right]$. From Lemma B.1,

$$\left[\boldsymbol{\beta} \mid \boldsymbol{y}^{tr}\right] \sim N_p\left(\boldsymbol{A}\boldsymbol{\nu} = \boldsymbol{\mu}_{\beta|tr}, \boldsymbol{A} = \boldsymbol{\Sigma}_{\beta|tr}\right), \tag{4.4.3}$$

where $\boldsymbol{\mu}_{\beta|tr}$ and $\boldsymbol{\Sigma}_{\beta|tr}$ are defined by (4.2.1) and (4.2.2) respectively. A similar but simpler argument applies to Case (b) and gives

$$\left[\boldsymbol{\beta} \mid \boldsymbol{y}^{tr}\right] \sim N_p\left(\left(\boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr}\right)^{-1} \times \left(\boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{y}^{tr}\right),\ \lambda_Z^{-1}\left(\boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr}\right)^{-1}\right).$$

For either Case (a) or (b), the remainder of the proof requires the evaluation of (4.4.1) whose integrand is proportional to

$$\begin{aligned}&\exp\left\{-\frac{1}{2}\left[\left(\boldsymbol{y}^{te} - \boldsymbol{D}_1\boldsymbol{\beta} - \boldsymbol{d}_2\right)^\top \boldsymbol{D}_3\left(\boldsymbol{y}^{te} - \boldsymbol{D}_1\boldsymbol{\beta} - \boldsymbol{d}_2\right) + \boldsymbol{\beta}^\top \boldsymbol{D}_4\boldsymbol{\beta} - 2\boldsymbol{d}_5^\top\boldsymbol{\beta}\right]\right\}\\ \propto\ &\exp\left\{-\frac{1}{2}(\boldsymbol{y}^{te})^\top \boldsymbol{D}_3\boldsymbol{y}^{te} + \boldsymbol{d}_6^\top\boldsymbol{y}^{te} + (\boldsymbol{y}^{te})^\top\boldsymbol{D}_7\boldsymbol{\beta} - \frac{1}{2}\boldsymbol{\beta}^\top\boldsymbol{D}_8\boldsymbol{\beta} + \boldsymbol{d}_9^\top\boldsymbol{\beta}\right\}\end{aligned}$$

where the matrices $\boldsymbol{D}_1$, $\boldsymbol{D}_3$, $\boldsymbol{D}_4$, $\boldsymbol{D}_7$, $\boldsymbol{D}_8$ and vectors $\boldsymbol{d}_2$, $\boldsymbol{d}_5$, $\boldsymbol{d}_6$, $\boldsymbol{d}_9$ are obtained from the terms in $\pi(\boldsymbol{y}^{te} \mid \boldsymbol{\beta}, \boldsymbol{y}^{tr})$ and $\pi(\boldsymbol{\beta} \mid \boldsymbol{y}^{tr})$ and all coefficients are functionally independent of $\boldsymbol{\beta}$ and $\boldsymbol{y}^{te}$. Thus

$$\begin{aligned}\pi\left(\boldsymbol{y}^{te} \mid \boldsymbol{y}^{tr}\right) &\propto \exp\left\{-\frac{1}{2}(\boldsymbol{y}^{te})^\top\boldsymbol{D}_3\boldsymbol{y}^{te} + \boldsymbol{d}_6^\top\boldsymbol{y}^{te}\right\}\\ &\qquad \times \int_{\mathbb{R}^p} \exp\left\{-\frac{1}{2}\boldsymbol{\beta}^\top\boldsymbol{D}_8\boldsymbol{\beta} + \left[(\boldsymbol{y}^{te})^\top\boldsymbol{D}_7 + \boldsymbol{d}_9^\top\right]\boldsymbol{\beta}\right\}\, d\boldsymbol{\beta}\\ &\propto \exp\left\{-\frac{1}{2}(\boldsymbol{y}^{te})^\top\boldsymbol{D}_3\boldsymbol{y}^{te} + \boldsymbol{d}_6^\top\boldsymbol{y}^{te}\right\}\\ &\qquad \times \exp\left\{\frac{1}{2}(\boldsymbol{y}^{te})^\top\boldsymbol{D}_7\boldsymbol{D}_8^{-1}\boldsymbol{D}_7^\top\boldsymbol{y}^{te} + \boldsymbol{d}_9^\top\boldsymbol{D}_8^{-1}\boldsymbol{D}_7^\top\boldsymbol{y}^{te}\right\}\\ &= \exp\left\{-\frac{1}{2}(\boldsymbol{y}^{te})^\top\boldsymbol{D}_{10}\,\boldsymbol{y}^{te} + \boldsymbol{d}_{11}^\top\boldsymbol{y}^{te}\right\}\end{aligned}$$

using (B.1.2) of Appendix B.1 to evaluate the integral, which shows that $\pi(\boldsymbol{y}^{te} \mid \boldsymbol{y}^{tr})$ is multivariate normally distributed with mean $\boldsymbol{\mu}_{te|tr} = \boldsymbol{D}_{10}^{-1}\boldsymbol{d}_{11}$ and covariance matrix $\boldsymbol{\Sigma}_{te|tr} = \boldsymbol{D}_{10}^{-1}$. Algebra recovers the expressions (4.2.4) and (4.2.5) for Case (a) and those described below (4.2.6) for Case (b). ■

Proof of Theorem 4.2

In spirit, the proof of Theorem 4.2 with $\boldsymbol{\vartheta} = (\boldsymbol{\beta}, \lambda_Z)$ is similar to that of Theorem 4.1 but requires substantially more algebraic manipulation because of the presence of λ_Z. Cases (a) and (b) of Theorem 4.2 consider three $[\boldsymbol{\beta}, \lambda_Z]$ priors. Only Case (a) will be discussed here, i.e.,

$$[\boldsymbol{\beta} \mid \lambda_Z] \sim N_p\left(\boldsymbol{b}_\beta, \lambda_Z^{-1}\boldsymbol{V}_\beta\right) \quad \text{and} \quad [\lambda_Z] \sim \Gamma(c, d)$$

with *known* $\boldsymbol{b}_\beta$, $\boldsymbol{V}_\beta$, c, and d. The remaining cases use similar arguments but are less complicated because the $[\boldsymbol{\beta}, \lambda_Z]$ prior is simpler.

Theorem 4.2 gives the predictive distribution of the n_e test outputs $\boldsymbol{y}^{te} = (y(\boldsymbol{x}_1^{te}), \ldots, y(\boldsymbol{x}_{n_e}^{te}))$ based on the n_s training outputs $\boldsymbol{y}^{tr} = (y(\boldsymbol{x}_1^{tr}), \ldots, y(\boldsymbol{x}_{n_s}^{tr}))$. It also provides the posterior distributions of λ_Z and $\boldsymbol{\beta}$ which summarize the information about these model parameters given the training data. To simplify the exposition below, some of the notation introduced in Theorem 4.2 is repeated, and several additional pieces of notation are defined.

- $\boldsymbol{G} = \boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1} \boldsymbol{F}_{tr}$,
- $\widehat{\boldsymbol{\beta}} = \boldsymbol{G}^{-1} \boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1} \boldsymbol{y}^{tr}$,
- $\boldsymbol{\Sigma}_\pi = \boldsymbol{G}^{-1} + \boldsymbol{V}_\beta$,
- $\boldsymbol{\Sigma}_{\beta|tr} = \left(\boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1} \boldsymbol{F}_{tr} + \boldsymbol{V}_\beta^{-1}\right)^{-1} = \left(\boldsymbol{G} + \boldsymbol{V}_\beta^{-1}\right)^{-1}$,
- $\boldsymbol{\mu}_{\beta|tr} = \left(\boldsymbol{G} + \boldsymbol{V}_\beta^{-1}\right)^{-1} \left(\boldsymbol{G}\widehat{\boldsymbol{\beta}} + \boldsymbol{V}_\beta^{-1}\boldsymbol{b}_\beta\right) = \boldsymbol{\Sigma}_{\beta|tr}\left(\boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1} \boldsymbol{y}^{tr} + \boldsymbol{V}_\beta^{-1}\boldsymbol{b}_\beta\right)$,
- $\mathrm{SS}_{tr} = \left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right)^\top \boldsymbol{R}_{tr}^{-1} \left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right)$, and
- $\mathrm{SS}_P = \left(\widehat{\boldsymbol{\beta}} - \boldsymbol{b}_\beta\right)^\top \boldsymbol{\Sigma}_\pi^{-1} \left(\widehat{\boldsymbol{\beta}} - \boldsymbol{b}_\beta\right)$.

As usual, let $(\boldsymbol{Y}^{te}, \boldsymbol{Y}^{tr})$ denote the process model for $(\boldsymbol{y}^{te}, \boldsymbol{y}^{tr})$. Recall that $(\boldsymbol{Y}^{te}, \boldsymbol{Y}^{tr})$ follows a two-stage hierarchical model with first-stage

$$\left[\begin{pmatrix}\boldsymbol{Y}^{te}\\ \boldsymbol{Y}^{tr}\end{pmatrix} \middle| \boldsymbol{\beta}, \lambda_Z\right] \sim N_{n_e+n_s}\left(\begin{pmatrix}\boldsymbol{F}_{te}\\ \boldsymbol{F}_{tr}\end{pmatrix}\boldsymbol{\beta},\ \lambda_Z^{-1}\begin{bmatrix}\boldsymbol{R}_{te} & \boldsymbol{R}_{te,tr}\\ \boldsymbol{R}_{te,tr}^\top & \boldsymbol{R}_{tr}\end{bmatrix}\right), \tag{4.4.4}$$

where $\boldsymbol{\beta}$ and λ_Z are *unknown* while the correlation parameters, $\boldsymbol{\kappa}$, are *known*. Just as the results in Theorem 4.1 should be regarded as conditional on λ_Z and $\boldsymbol{\kappa}$, the calculations below are conditional on the correlation parameters. This fact was used in Sect. 4.3.

In overview, the conditional densities $\pi(\lambda_Z \mid \boldsymbol{y}^{tr})$ and $\pi(\boldsymbol{\beta} \mid \boldsymbol{y}^{tr})$ are computed by finding the conditional density $\pi(\boldsymbol{\beta}, \lambda_Z \mid \boldsymbol{y}^{tr})$ and calculating the marginals

$$\pi\left(\lambda_Z \,\middle|\, \boldsymbol{y}^{tr}\right) = \int_{\mathbb{R}^p} \pi\left(\boldsymbol{\beta}, \lambda_Z \,\middle|\, \boldsymbol{y}^{tr}\right) d\boldsymbol{\beta} \quad \text{and}$$

$$\pi\left(\boldsymbol{\beta} \,\middle|\, \boldsymbol{y}^{tr}\right) = \int_0^\infty \pi\left(\boldsymbol{\beta}, \lambda_Z \,\middle|\, \boldsymbol{y}^{tr}\right) d\lambda_Z. \tag{4.4.5}$$

As for Theorem 4.2, this task is simplified because only terms in $\pi(\boldsymbol{\beta}, \lambda_Z \mid \boldsymbol{y}^{tr})$ involving $\boldsymbol{\beta}$ and λ_Z need be determined in order to perform the integrations in (4.4.5) and hence to determine the posterior density kernels.

Start with

$$\pi\left(\boldsymbol{\beta}, \lambda_Z \,\middle|\, \boldsymbol{y}^{tr}\right) \propto \pi\left(\boldsymbol{y}^{tr} \,\middle|\, \boldsymbol{\beta}, \lambda_Z\right) \times \pi\left(\boldsymbol{\beta}, \lambda_Z\right)$$

and recognize that the kernel of $\pi(\boldsymbol{y}^{tr} \mid \boldsymbol{\beta}, \lambda_Z)$ is immediate from (4.4.4) as

$$\pi\left(\boldsymbol{y}^{tr} \,\middle|\, \boldsymbol{\beta}, \lambda_Z\right) \propto \lambda_Z^{n_s/2} \, \exp\left\{-\frac{\lambda_Z}{2}\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\boldsymbol{\beta}\right)^\top \boldsymbol{R}_{tr}^{-1}\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\boldsymbol{\beta}\right)\right\},$$

while the joint prior is

$$\pi\left(\boldsymbol{\beta}, \lambda_Z\right) \propto \lambda_Z^{p/2} \, \exp\left\{-\frac{\lambda_Z}{2}\left(\boldsymbol{\beta} - \boldsymbol{b}_\beta\right)^\top \boldsymbol{V}_\beta^{-1}\left(\boldsymbol{\beta} - \boldsymbol{b}_\beta\right)\right\} \times \lambda_Z^{c-1} \exp\left\{-d\lambda_Z\right\}.$$

Calculation gives

$$\pi\left(\boldsymbol{\beta}, \lambda_Z \,\middle|\, \boldsymbol{y}^{tr}\right) \propto \lambda_Z^{(2c+n_s+p)/2-1} \, \exp\left\{-\frac{\lambda_Z}{2}\left(2d + Q_1 + Q_2\right)\right\}$$

where

$$Q_1 = \left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\boldsymbol{\beta}\right)^\top \boldsymbol{R}_{tr}^{-1}\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\boldsymbol{\beta}\right)$$

and

$$Q_2 = \left(\boldsymbol{\beta} - \boldsymbol{b}_\beta\right)^\top \boldsymbol{V}_\beta^{-1}\left(\boldsymbol{\beta} - \boldsymbol{b}_\beta\right).$$

The key to computing (4.4.5) is to show that $Q_1 + Q_2$ can be rewritten as

$$\mathrm{SS}_{tr} + \mathrm{SS}_P + \left(\boldsymbol{\beta} - \boldsymbol{\mu}_{\beta|tr}\right)^\top \boldsymbol{\Sigma}_{\beta|tr}^{-1}\left(\boldsymbol{\beta} - \boldsymbol{\mu}_{\beta|tr}\right), \tag{4.4.6}$$

where SS_{tr} and SS_P are defined in the list of symbols on page 138. Notice that SS_{tr} and SS_P are *independent* of $\boldsymbol{\beta}$, while the third term is a quadratic form in $\boldsymbol{\beta}$. Hence from formulas (4.4.5) and (4.4.6),

$$\begin{aligned}\pi\left(\lambda_Z \,\middle|\, \boldsymbol{y}^{tr}\right) \propto\, & \lambda_Z^{(2c+n_s+p)/2-1} \\ & \times \exp\left\{-\frac{\lambda_Z}{2}\left(2d + \mathrm{SS}_{tr} + \mathrm{SS}_P\right)\right\} \\ & \times \int_{\mathbb{R}^p} \exp\left\{-\frac{1}{2}\left(\boldsymbol{\beta} - \boldsymbol{\mu}_{\beta|tr}\right)^\top \left(\frac{1}{\lambda_Z}\boldsymbol{\Sigma}_{\beta|tr}\right)^{-1}\left(\boldsymbol{\beta} - \boldsymbol{\mu}_{\beta|tr}\right)\right\} d\boldsymbol{\beta}.\end{aligned}$$

From the density of the multivariate normal distribution, formula (B.1.3) shows

$$\begin{aligned}\int_{\mathbb{R}^p} \exp\left\{-\frac{1}{2}\left(\boldsymbol{\beta} - \boldsymbol{\mu}_{\beta|tr}\right)^\top \left(\frac{1}{\lambda_Z}\boldsymbol{\Sigma}_{\beta|tr}\right)^{-1}\left(\boldsymbol{\beta} - \boldsymbol{\mu}_{\beta|tr}\right)\right\} d\boldsymbol{\beta} &\propto \left(\det\left(\frac{1}{\lambda_Z}\boldsymbol{\Sigma}_{\beta|tr}\right)\right)^{1/2} \\ &\propto \frac{1}{(\lambda_Z)^{p/2}}.\end{aligned}$$

This yields

$$\begin{aligned}\pi\left(\lambda_Z \,\middle|\, \boldsymbol{y}^{tr}\right) &\propto \lambda_Z^{(2c+n_s+p)/2-1} \times \exp\left\{-\frac{\lambda_Z}{2}\left(2d + \mathrm{SS}_{tr} + \mathrm{SS}_P\right)\right\} \times \frac{1}{(\lambda_Z)^{p/2}} \\ &\propto \lambda_Z^{(2c+n_s)/2-1} \exp\left\{-\lambda_Z \frac{2d + \mathrm{SS}_{tr} + \mathrm{SS}_P}{2}\right\}. \end{aligned} \tag{4.4.7}$$

Recognizing that Eq. (4.4.7) is the kernel of the gamma density with shape parameter $c + n_s/2$ and rate parameter $(2d + \mathrm{SS}_{tr} + \mathrm{SS}_P)/2$ confirms the λ_Z posterior stated in Theorem 4.2.

Analogously, the $\boldsymbol{\beta}$ posterior can be computed by applying (4.4.6) to the second equation of (4.4.5) producing

$$\pi\left(\boldsymbol{\beta} \,\middle|\, \boldsymbol{y}^{tr}\right) \propto \int_0^\infty \lambda_Z^{(2c+n_s+p)/2-1} \times \exp\left\{-\lambda_Z \frac{2d + \mathrm{SS}_{tr} + \mathrm{SS}_P + \left(\boldsymbol{\beta} - \boldsymbol{\mu}_{\beta|tr}\right)^\top \boldsymbol{\Sigma}_{\beta|tr}^{-1} \left(\boldsymbol{\beta} - \boldsymbol{\mu}_{\beta|tr}\right)}{2}\right\} d\lambda_Z$$

which can be integrated using

$$\int_0^\infty e^{-\beta w} w^{\alpha-1} dw = \frac{\Gamma(\alpha)}{\beta^\alpha}.$$

Omitting terms that do not depend on $\boldsymbol{\beta}$ or λ_Z gives

$$\begin{aligned}\pi\left(\boldsymbol{\beta} \,\middle|\, \boldsymbol{y}^{tr}\right) &\propto \left[2d + \mathrm{SS}_{tr} + \mathrm{SS}_P + \left(\boldsymbol{\beta} - \boldsymbol{\mu}_{\beta|tr}\right)^\top \boldsymbol{\Sigma}_{\beta|tr}^{-1} \left(\boldsymbol{\beta} - \boldsymbol{\mu}_{\beta|tr}\right)\right]^{-(2c+n_s+p)/2} \\ &\propto \left[1 + \frac{(2c+n_s)}{(2d + \mathrm{SS}_{tr} + \mathrm{SS}_P)} \times \frac{\left(\boldsymbol{\beta} - \boldsymbol{\mu}_{\beta|tr}\right)^\top \boldsymbol{\Sigma}_{\beta|tr}^{-1} \left(\boldsymbol{\beta} - \boldsymbol{\mu}_{\beta|tr}\right)}{(2c+n_s)}\right]^{-(2c+n_s+p)/2}. \end{aligned} \tag{4.4.8}$$

Comparing the kernel (4.4.8) with (B.4.1) shows $\pi(\boldsymbol{\beta} \mid \boldsymbol{y}^{tr})$ has the p-variate t density with $2c+n_s$ degrees of freedom, location parameter $\boldsymbol{\mu}_{\beta|tr}$, and scale matrix $(2d+\mathrm{SS}_{tr}+\mathrm{SS}_P)\boldsymbol{\Sigma}_{\beta|tr}/(2c+n_s)$ as stated in Theorem 4.2.

Lastly, the predictive density of $\boldsymbol{Y}^{te}$ can be obtained from

$$\begin{aligned}\pi\left(\boldsymbol{y}^{te} \,\middle|\, \boldsymbol{y}^{tr}\right) &= \int\int \pi\left(\boldsymbol{y}^{te}, \boldsymbol{\beta}, \lambda_Z \,\middle|\, \boldsymbol{y}^{tr}\right) d\boldsymbol{\beta}\, d\lambda_Z \\ &= \int\int \pi\left(\boldsymbol{y}^{te} \,\middle|\, \boldsymbol{y}^{tr}, \boldsymbol{\beta}, \lambda_Z\right) \times \pi\left(\boldsymbol{\beta} \,\middle|\, \boldsymbol{y}^{tr}, \lambda_Z\right) d\boldsymbol{\beta} \\ &\qquad \times \pi\left(\lambda_Z \,\middle|\, \boldsymbol{y}^{tr}\right) d\lambda_Z\,. \end{aligned}$$

From (4.4.2), $\pi(\boldsymbol{y}^{te} \mid \boldsymbol{y}^{tr}, \boldsymbol{\beta}, \lambda_Z)$ is n_e-variate multivariate normal with mean $\boldsymbol{\mu}_{te|tr,\beta} = \boldsymbol{F}_{te}\boldsymbol{\beta} + \boldsymbol{R}_{te,tr}\boldsymbol{R}_{tr}^{-1}(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\boldsymbol{\beta})$ and covariance $\boldsymbol{\Sigma}_{te|tr,\beta} = \lambda_Z^{-1}\left(\boldsymbol{R}_{te} - \boldsymbol{R}_{te,tr}\boldsymbol{R}_{tr}^{-1}\boldsymbol{R}_{te,tr}^\top\right)$. The density $\pi(\boldsymbol{\beta} \mid \boldsymbol{y}^{tr}, \lambda_Z)$ is given by (4.4.3) taking $\lambda_\beta = \lambda_Z$, while, from (4.4.7), the posterior $\pi(\lambda_Z \mid \boldsymbol{y}^{tr})$ is gamma with shape parameter $c + n_s/2$ and rate parameter $(2d + \mathrm{SS}_{tr} + \mathrm{SS}_P)/2$. Thus the integral is based on terms whose forms are similar to those used in the λ_Z and $\boldsymbol{\beta}$ posterior calculations. Algebra again gives the result.

To provide some of the algebraic details required to prove (4.4.6),

$$\begin{aligned} Q_1 &= \left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\boldsymbol{\beta}\right)^\top \boldsymbol{R}_{tr}^{-1} \left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\boldsymbol{\beta}\right) \\ &= \left(\left[\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right] + \left[\boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}} - \boldsymbol{F}_{tr}\boldsymbol{\beta}\right]\right)^\top \boldsymbol{R}_{tr}^{-1} \left(\left[\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right] + \left[\boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}} - \boldsymbol{F}_{tr}\boldsymbol{\beta}\right]\right) \\ &= \left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right)^\top \boldsymbol{R}_{tr}^{-1} \left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right) + \left(\boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}} - \boldsymbol{F}_{tr}\boldsymbol{\beta}\right)^\top \boldsymbol{R}_{tr}^{-1} \left(\boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}} - \boldsymbol{F}_{tr}\boldsymbol{\beta}\right) \\ &\quad + 2\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right)^\top \boldsymbol{R}_{tr}^{-1} \left(\boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}} - \boldsymbol{F}_{tr}\boldsymbol{\beta}\right) \\ &= \mathrm{SS}_{tr} + \left(\boldsymbol{\beta} - \widehat{\boldsymbol{\beta}}\right)^\top \boldsymbol{G}\left(\boldsymbol{\beta} - \widehat{\boldsymbol{\beta}}\right) + 0 \end{aligned} \tag{4.4.9}$$

by noting that the cross product term is zero, i.e.,

$$\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}}\right)^\top \boldsymbol{R}_{tr}^{-1} \left(\boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}} - \boldsymbol{F}_{tr}\boldsymbol{\beta}\right) = \\ (\boldsymbol{y}^{tr})^\top (\boldsymbol{I}_{n_s} - \boldsymbol{F}_{tr}\boldsymbol{G}^{-1}\boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1})^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr}(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}) = 0\,.$$

Fixing SS_{tr} and expanding the individual $\widehat{\boldsymbol{\beta}}$, $\boldsymbol{\beta}$, and $\boldsymbol{b}_\beta$ terms in (4.4.9) and also in Q_2 give

$$\begin{aligned} Q_1 + Q_2 &= \left(\mathrm{SS}_{tr} + \left(\boldsymbol{\beta} - \widehat{\boldsymbol{\beta}}\right)^\top \boldsymbol{G}\left(\boldsymbol{\beta} - \widehat{\boldsymbol{\beta}}\right)\right) + \left(\boldsymbol{\beta} - \boldsymbol{b}_\beta\right)^\top \boldsymbol{V}_\beta^{-1}\left(\boldsymbol{\beta} - \boldsymbol{b}_\beta\right) \\ &= \mathrm{SS}_{tr} + \boldsymbol{b}_\beta^\top \boldsymbol{V}_\beta^{-1}\boldsymbol{b}_\beta + \widehat{\boldsymbol{\beta}}^\top \boldsymbol{G}\widehat{\boldsymbol{\beta}} + \boldsymbol{\beta}^\top \boldsymbol{\Sigma}_{\beta|tr}^{-1}\boldsymbol{\beta} - 2\boldsymbol{\beta}^\top \boldsymbol{\Sigma}_{\beta|tr}^{-1}\boldsymbol{\mu}_{\beta|tr} && (4.4.10) \\ &= \mathrm{SS}_{tr} + \boldsymbol{b}_\beta^\top \boldsymbol{V}_\beta^{-1}\boldsymbol{b}_\beta + \widehat{\boldsymbol{\beta}}^\top \boldsymbol{G}\widehat{\boldsymbol{\beta}} \\ &\quad + \left(\boldsymbol{\beta} - \boldsymbol{\mu}_{\beta|tr}\right)^\top \boldsymbol{\Sigma}_{\beta|tr}^{-1}\left(\boldsymbol{\beta} - \boldsymbol{\mu}_{\beta|tr}\right) - \boldsymbol{\mu}_{\beta|tr}^\top \boldsymbol{\Sigma}_{\beta|tr}^{-1}\boldsymbol{\mu}_{\beta|tr} && (4.4.11) \\ &= \mathrm{SS}_{tr} + \left(\boldsymbol{\beta} - \boldsymbol{\mu}_{\beta|tr}\right)^\top \boldsymbol{\Sigma}_{\beta|tr}^{-1}\left(\boldsymbol{\beta} - \boldsymbol{\mu}_{\beta|tr}\right) \\ &\quad + \left(\widehat{\boldsymbol{\beta}} - \boldsymbol{b}_\beta\right)^\top \boldsymbol{\Sigma}_\pi^{-1}\left(\widehat{\boldsymbol{\beta}} - \boldsymbol{b}_\beta\right), && (4.4.12) \end{aligned}$$

where (4.4.11) follows by adding and subtracting $\boldsymbol{\mu}_{\beta|tr}^\top \boldsymbol{\Sigma}_{\beta|tr}^{-1}\boldsymbol{\mu}_{\beta|tr}$ to (4.4.10) and (4.4.12) follows from (4.4.11) by substituting

$$\boldsymbol{\mu}_{\beta|tr}^\top \boldsymbol{\Sigma}_{\beta|tr}^{-1}\boldsymbol{\mu}_{\beta|tr} = \boldsymbol{b}_\beta^\top \boldsymbol{V}_\beta^{-1}\boldsymbol{b}_\beta + \widehat{\boldsymbol{\beta}}^\top \boldsymbol{G}\widehat{\boldsymbol{\beta}} - \left(\widehat{\boldsymbol{\beta}} - \boldsymbol{b}_\beta\right)^\top \boldsymbol{\Sigma}_\pi^{-1}\left(\widehat{\boldsymbol{\beta}} - \boldsymbol{b}_\beta\right). \tag{4.4.13}$$

Identity (4.4.13) can be shown by expanding the left hand as

$$\boldsymbol{\mu}_{\beta|tr}^\top \boldsymbol{\Sigma}_{\beta|tr}^{-1}\boldsymbol{\mu}_{\beta|tr} = \left(\boldsymbol{G}\widehat{\boldsymbol{\beta}} + \boldsymbol{V}_\beta^{-1}\boldsymbol{b}_\beta\right)^\top \boldsymbol{\Sigma}_{\beta|tr}\left(\boldsymbol{G}\widehat{\boldsymbol{\beta}} + \boldsymbol{V}_\beta^{-1}\boldsymbol{b}_\beta\right),$$

applying the definition of $\boldsymbol{\mu}_{\beta|tr}$, and then expanding this quadratic form into three terms. Now simplify the quadratic term in $\boldsymbol{b}_\beta$ using $\boldsymbol{\Sigma}_{\beta|tr} = \left(\boldsymbol{V}_\beta^{-1} + \boldsymbol{G}\right)^{-1} = \boldsymbol{V}_\beta -$

$\boldsymbol{V}_\beta \left(\boldsymbol{V}_\beta + \boldsymbol{G}^{-1}\right)^{-1} \boldsymbol{V}_\beta$ and the quadratic term in $\widehat{\boldsymbol{\beta}}$ using $\left(\boldsymbol{V}_\beta^{-1} + \boldsymbol{G}\right)^{-1} = \boldsymbol{G}^{-1} - \boldsymbol{G}^{-1}\left(\boldsymbol{V}_\beta + \boldsymbol{G}^{-1}\right)^{-1} \boldsymbol{G}^{-1}$. Apply $\left(\boldsymbol{V}_\beta + \boldsymbol{G}^{-1}\right)^{-1} = \boldsymbol{V}_\beta^{-1}\left(\boldsymbol{V}_\beta^{-1} + \boldsymbol{G}\right)^{-1} \boldsymbol{G}$ to the cross term to show it is equal to $2\boldsymbol{b}_\beta^\top \boldsymbol{\Sigma}_\pi^{-1} \widehat{\boldsymbol{\beta}}$. The resulting sum simplifies to the right-hand side of (4.4.13).

4.4.2 Eliciting Priors for Bayesian Regression

How can prior parameters be selected in a Bayesian analysis? Generic advice is given in Gelman et al. (2013) and specific advice for the hierarchical regression + stationary GP in Oakley (2002). Ideally, one should base prior parameter choices on both expert opinion and external simulator output for the same or a similar physical system to the $y(\boldsymbol{x})$ being emulated. Both expert opinion and especially the examination of external simulator data should focus on the *observable characteristics* of the output. These characteristics should include, among other $y(\boldsymbol{x})$ features: its maximum and minimum values, its range, the number of local maxima and minima, the relative activity of the inputs, and slope information as specific inputs vary.

4.4.3 Alternative Sampling Algorithms

Section 4.3 has emphasized the use of the Metropolis–Hastings algorithm, the Gibbs algorithm, and a Gibbs algorithm that includes MH draws for steps where the conditional may not be known. For example, the latter would typically be the case for the regression + stationary GP model when drawing correlation parameters from the conditional of the correlation parameters given the data, the regression coefficients, and the precision parameters.

There have been a number of alternatives to the Metropolis–Hastings algorithm that have been used in the computer experiments literature. We point only to the Delayed Rejection Adaptive Metropolis (DRAM) algorithm of Haario et al. (2006) and the overview tutorial article Andrieu and Thoms (2008) as a starting place for interested readers.

4.4.4 Software for Computing Bayesian Predictions

The authors of this volume provide a partial list of software for fitting the Bayesian GP models described in this chapter, without endorsing any particular package. The majority of the programs below use the Gaussian correlation function. As always, the authors make no claim of completeness of the list nor are the comments a review of the accuracy and features of the programs. Indeed, many of them are in continuous development, and web searches will provide up-to-date information about any particular program as well as others that have been developed since the publication of this book.

1. GPMSA (Gaussian Process Models for Simulation Analysis) is a MATLAB program that provides Bayesian prediction for both univariate and multivariate simulator output: http://go.osu.edu/GPMSA
2. tgp is a R package for fitting Bayesian Treed GP Models
3. Stan is a programming language that implements full Bayesian statistical inference; it is coded in C++ with both R language and MATLAB interfaces: http://mc-stan.org
4. The Gaussian Processes Web Site lists both references and software for fitting hierarchical GP models: http://www.gaussianprocess.org.
5. Dakota is a software package developed at Sandia National Laboratories for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis. Bayesian emulation (prediction and uncertainty quantification) of computer simulation models has been added to recent versions of Dakota: http://dakota.sandia.gov
6. MUCM (Managing Uncertainty in Complex Models) is a resource concerned with issues of uncertainty in computer simulation models including uncertainty quantification, uncertainty propagation, uncertainty analysis, sensitivity analysis, and calibration. The website http://www.mucm.ac.uk/Pages/MCSGSoftware.html contains a list of Bayesian and frequentist software for prediction, calibration, and other Bayesian procedures.
7. SHELF (SHeffield ELicitation Framework) is a package of documents, templates, and software to carry out elicitation of probability distributions for uncertain quantities from a group of experts: http://tonyohagan.co.uk/shelf.

Chapter 5
Space-Filling Designs for Computer Experiments

5.1 Introduction

This chapter and the next discuss how to select inputs at which to compute the output of a computer experiment to achieve specific goals. The inputs one selects constitute the "experimental design." As in previous chapters, the inputs are referred to as "runs." The region corresponding to the values of the inputs that is to be studied is called the *experimental region*. A point in this region corresponds to a specific set of values of the inputs. Thus, an experimental design is a specification of points (runs) in the experimental region at which the response is to be computed.

This chapter begins by reviewing some of the basic principles of classical experimental design and then presents an overview of some of the strategies that have been employed in computer experiments. For details concerning classical design see, for example, the books by Atkinson and Donev (1992), Box and Draper (1987), Dean et al. (2017), Pukelsheim (1993), Silvey (1980), and Wu and Hamada (2009).

5.1.1 Some Basic Principles of Experimental Design

Suppose that one observes a response and wishes to study how that response varies as one changes a set of (potentially) explanatory inputs ("factors"). In physical experiments, there are a number of issues that can make such a study problematic. First, the response may be affected by factors other than the inputs that have been selected for study. Unless one can completely control the effects of these additional factors, repeated observations at the same values of the inputs will vary as these additional factors change. The effects of additional factors can either be *unsystematic* (random) or *systematic*. Unsystematic effects are usually referred to as random error (or as "measurement error" or as "noise"). Systematic effects are often referred to as bias. There are strategies for dealing with both random error and bias.

Replication and *blocking* are two techniques used to estimate and control the magnitude of random error. Replication (observing the response multiple times at the same set of inputs) allows one to directly estimate the magnitude and distribution of random error. Also, the sample means of replicated responses have smaller variances than the individual responses. Thus, the relation between these means and the inputs gives a clearer picture of the effects of the inputs because uncertainty from random error is reduced. In general, the more observations one has, the more information one has about the relation between the response and the inputs.

Blocking involves sorting experimental material into, or running the experiment in, relatively homogeneous groups called blocks. The corresponding analysis explores the relation between the response and the inputs within blocks and then combines the results across blocks. Because of the homogeneity within a block, the observed random error is less *within* a block than *between* blocks and the effects of the inputs more easily seen. There is an enormous body of literature on block designs, including both statistical and combinatorial issues. General discussions include John (1980), John (1987), Raghavarao (1971), or Street and Street (1987).

Bias is typically controlled by *randomization* and by exploring how the response changes as the inputs change. Randomization is accomplished by using a well-defined chance mechanism to assign the input values as well as any other factors that may affect the response and that are under the control of the experimenter, such as the order of experimentation, to experimental material. Factors assigned at random to experimental material will not systematically affect the response. By basing inferences on changes in the response as the input changes, bias effects "cancel," at least on average.

Replication, blocking, and randomization are basic principles of experimental design for controlling random error and bias. However, random error and bias are not the only problems that face experimenters. Another problem occurs when one is interested in studying the effects of several inputs simultaneously and the input selections are (inadvertently) highly correlated. This sometimes occurs in observational studies. If, for example, the observed values of two inputs are positively correlated so that they increase together simultaneously, then it is difficult to distinguish their effects on the response. Was it the increase in just one or some combination of both that produced the observed change in the response? This problem is sometimes referred to as *collinearity*. Using an *orthogonal design* can overcome this problem. In an orthogonal design, the values of the inputs at which the response is observed are uncorrelated. An orthogonal design allows one to independently assess the effects of the different inputs. There is a large body of literature on finding orthogonal designs, generally in the context of factorial experiments. See, for example, Hedayat et al. (1999).

Another problem that can be partly addressed (or at least detected) by careful choice of an experimental design occurs when the assumptions one makes about the nature of the relation between the response and the inputs (the statistical model) are incorrect. For example, suppose one assumes that the relationship between the response and a single input is essentially linear when, in fact, it is highly nonlinear. Inferences based on the assumption that the relationship is linear will be incorrect.

It is important to be able to detect strong nonlinearities, and one will need to observe the response with at least three different values of the input in order to do so. The error that arises because the assumed model is incorrect is sometimes referred to as *model bias*. Diagnostics, such as scatterplots and quantile plots, are used to detect model bias. The ability to detect model bias is improved by careful choice of an experimental design, for example, by observing the response at a wide variety of values of the inputs. In general, one would like to select designs that enable one to detect violations of the fitted model and lead to inferences that are relatively insensitive to model bias. This usually requires specifying both the model one intends to fit to the data as well as the form of an alternative model whose bias one wishes to guard against; thus designs for model bias are selected to protect against certain types of bias. Box and Draper (1987) discuss this issue in more detail.

In addition to general principles, such as replication, blocking, randomization, orthogonality, and the ability to detect model bias, there exist very formal approaches to selecting an experimental design. The underlying principle is to consider the purpose of the experiment and the statistical model for the data and choose the design accordingly. If one can formulate the purpose of the experiment in terms of optimizing a particular quantity, one can then ask what inputs one should observe the response at to optimize this quantity. For example, if one is fitting a straight line to data, one might wish to select the design so as to give the most precise (minimum variance) estimate of the slope. This approach to selection of an experimental design is often referred to as *optimal design*. See Atkinson and Donev (1992), Pukelsheim (1993), or Silvey (1980) for more on the theory of optimal design. In the context of the linear model, popular criteria involve minimizing some function of the covariance matrix of the least squares estimates of the parameters. Some common functions are the determinant of the covariance matrix (the generalized variance), the trace of the covariance matrix (the average variance), and the average of the variance of the predicted response over the experimental region. A design minimizing the first criterion is called *D-optimal*, a design minimizing the second is called *A-optimal*, and a design minimizing the third is called *I-optimal*. In many experiments, especially experiments with multiple objectives, it may not be clear how to formulate the experiment goal in terms of some quantity that can be optimized. Furthermore, even if one can formulate the problem in this way, finding the optimal design may be quite difficult.

In many experiments all the inputs at which one will observe the response are specified in advance. These are sometimes referred to as a single-stage or one-stage experimental designs. However, there are good reasons for running experiments in multiple stages. Box et al. (1978) (page 303), advocate the use of sequential or multistage designs:

> In exploring a functional relationship it might appear reasonable at first sight to adopt a comprehensive approach in which the entire range of every factor was investigated. The resulting design might contain all combinations of several levels of all factors. However, when runs can be made in successive groups, this is an inefficient way to organize experimental programs. The situation relates to the paradox that the best time to design an experiment is after it is finished, the converse of which is that the worst time is at the beginning, when the least is known. If the entire experiment was designed at the outset, the following would have

to be assumed known: (1) which variables were the most important, (2) over what ranges the variables should be studied, (3) in what metrics the variables and responses should be considered (e.g., linear, logarithmic, or reciprocal scales), and (4) what multivariable transformations should be made (perhaps the effects of variables x_1 and x_2 would be most simply expressed in terms of their ratio x_1/x_2 and their sum $x_1 + x_2$).

The experimenter is least able to answer such questions at the outset of an investigation but gradually becomes more able to do so as a program evolves.

All the above arguments point to the desirability of a sequence of moderately sized designs and reassessment of the results as each group of experiments becomes available.

5.1.2 Design Strategies for Computer Experiments

Computer experiments based on simulator codes differ from traditional physical experiments in that repeated observations at the same set of inputs yield, aside from numerical error, identical responses. Hence replication is unnecessary. The greatest uncertainty in running a computer experiment arises because one only knows the exact form of the input-response relationship of the simulator code at the inputs of the experimental design (although the response can be computed at any additional input). Functional models that are used to describe and extend the simulator-determined relationship to parts of the experimental region where runs have *not* been made are *approximations* to the true relationship. The discrepancy between the response produced by the simulator code and the response predicted by an approximating model, both run at the same input, is the error in the approximating model. In the previous subsection such error was referred to as model bias.

Based on these observations, two principles for selecting designs in the types of computer experiments considered here are the following:

1. *Designs should not take more than one observation at any set of inputs. (But note that this principle assumes the computer code remains unchanged over time. When a design is run sequentially and the computer code is written and executed by a third party, it may be good policy to duplicate one of the design points in order to verify that the code has not been changed over the course of the experiment.)*
2. *Because one does not know the true relationship between the response and inputs, designs should allow one to fit a variety of models and should provide information about all portions of the experimental region.*

If a priori one believes that interesting features of the true model are just as likely to be in one part of the experimental region as another, if one's goal is to be able to do prediction over the entire range of the inputs and if one is running a single-stage experiment, it is plausible to use designs that spread the points (inputs, runs) at which one observes the response evenly throughout the region.

There are a number of ways to define what it means to spread points evenly throughout a region and these lead to various types of designs. This chapter discusses a number of these design principles. Among the designs considered are designs based on selecting points in the experimental region by certain sampling methods; designs based on measures of distance between points that allow one to quantify how evenly spread points are; designs based on measures of how close points are to being uniformly distributed throughout a region; and designs that are a hybrid of or variation on these designs. All the designs in this chapter will be referred to as *space-filling* or *exploratory* designs.

The term "space-filling" is used widely in the literature on computer experiments. It seems that in most cases, space-filling is meant in an intuitive sense and as a synonym for "evenly spread." However, it also has a more technical meaning. It can refer to a method for generating designs for any run size n_s, such that as n_s increases, the method produces designs that are increasingly dense in the design space (in other words, fill the design space). Vazquez and Bect (2011) analyze the limiting properties of the prediction variance for such designs. In particular, assuming the GP model is correct, for sufficiently large sample sizes, Vazquez and Bect (2011) show that no design will outperform certain space-filling designs (those with an asymptotic fill distance of $O(1/n_s^{1/d})$) in terms of the rate at which the maximum of the mean square prediction error decreases as n_s increases. This provides some theoretical justification for using space-filling designs.

When runs of a computer experiment are expensive or time-consuming so that observing the response at a "large" number of inputs is not possible, what is a reasonable sample size that will allow one to fit the models described in Chaps. 2–4? One early rule-of-thumb suggested by Chapman et al. (1994) and Jones et al. (1998) is to use a sample size of $10d$ when the input space is of dimension d. However, because the "volume" of the design space increases as a power of d, $10d$ points becomes a very sparse sample as d increases. Obviously ten points evenly spread over the unit interval are much more densely distributed than 100 points in the ten-dimensional unit cube. So is the $10d$ rule of thumb reasonable? Loeppky et al. (2009) carefully investigate this issue and conclude that a sample size of $10d$ is a reasonable rule-of-thumb for an initial experiment when $d \leq 5$. When the response is sensitive to relatively few of the inputs, the rule is also reasonable for an initial experiment for d up to 20 or even larger. Loeppky et al. (2009) also discuss diagnostics one can use to determine whether additional observations are needed (beyond those recommended by the $10d$ rule-of-thumb) and approximately how many might be needed to improve overall fit. They point out that one should always check the accuracy of the predictor fitted to the data and if it is poor, additional observations (perhaps many) may be needed.

In the spirit of Loeppky et al. (2009), a rough argument about the adequacy of the $10d$ rule-of-thumb is as follows. Suppose that polynomial models provide a reasonable class of approximations for a set of simulator code input–output relationships. The minimum number of points needed to uniquely determine a response surface of order r in d variables (including all monomials of order r or less) is

$$\binom{r+d}{r}. \tag{5.1.1}$$

For a second-order response surface ($r = 2$), the $10d$ rule-of-thumb exceeds equation (5.1.1) up to $d = 16$. For a third-order response surface, the $10d$ rule-of-thumb exceeds Eq. (5.1.1) only up to $d = 4$. For a fourth-order response surface, the $10d$ rule-of-thumb is greater than Eq. (5.1.1) only for $d = 2$. Also, for an input–output relation such as $y = \sin(c\pi x), 0 \leq x \leq 1$, the $10d$ rule won't allow for enough observations in one dimension to produce an adequate predictor for large c, assuming one has no prior knowledge of the functional form of this relationship.

While many real-life applications are not as complicated as those suggested in the previous paragraph, there are cases when using an experimental design of size $10d$ can be inadequate. For example, Chen et al. (2011) discuss a simulator experiment concerning bistable laser diodes in which the $d = 2$ response surface is quite rough over a portion of the design space and would require substantially more than 20 ($10d$) observations to accurately approximate.

In practice, the true model that describes the relation between the inputs and the response is unknown. However, if the models to be fit to the data come from a sufficiently broad class, one may be willing to assume some model in this class is (to good approximation) "correct." In this case it is possible to formulate specific criteria for choosing a design and adopt an optimal design approach. Because the models considered in the previous chapters are remarkably flexible, this approach seems reasonable for these models. Thus, Chap. 6 discusses some criterion-based methods for selecting designs.

5.2 Designs Based on Methods for Selecting Random Samples

In the language of Sect. 1.1, the designs described in this section are used in cases when all components of $\boldsymbol{x}$ are control inputs as well as in cases when $\boldsymbol{x}$ contains both control and environmental inputs. However, most of these designs were originally motivated by their usefulness in applications where the inputs were all environmental variables; in this case the inputs are denoted by $\boldsymbol{X}$ to emphasize their random nature. Let $y(\cdot)$ denote the output of the code. When the inputs are environmental variables, the most comprehensive objective would be to find the distribution of the random variable $Y = y(\boldsymbol{X})$ when $\boldsymbol{X}$ has a known distribution. If, as is often the case, this is deemed too difficult, the easier problem of determining some aspect of its distribution such as its mean $E[Y] = \mu$ or its variance is considered. Several of the designs introduced in this section, in particular the Latin hypercube design, were developed to solve the problem of estimating μ in such a setting. However, the reader should bear in mind that such designs are useful in more general input settings.

5.2.1 Designs Generated by Elementary Methods for Selecting Samples

Intuitively, one would like designs for computer experiments to be space-filling when prediction accuracy over the entire experimental region is of primary interest. The reason for this is that interpolators are used as predictors (e.g., the BLUP or its Bayesian counterparts such as those that arise as the means of the predictive distributions derived in Chap. 4). Hence, the prediction error at any new input site is a function of its location relative to the design points. Indeed, Sect. 3.2 showed that the prediction error is *zero* at each of the design points. For this reason, designs that are not space-filling, for example, designs that concentrate points on the boundary of the design space, can yield predictors that perform quite poorly in portions of the experimental region that are sparsely observed.

Deterministic strategies for selecting the values of the inputs at which to observe the response choose values so they are spread evenly throughout or fill the experimental region. There are several methods that might be used to accomplish this, depending on what one means by "spreading points evenly" or "filling the experimental region."

A very simple strategy is to select points according to a regular grid pattern superimposed on the experimental region. For example, suppose the experimental region is the unit square $[0,1]^2 = [0,1] \times [0,1]$. If one wishes to observe the response at 25 evenly spaced points, one might consider the grid of points $\{0.1, 0.3, 0.5, 0.7, 0.9\} \times \{0.1, 0.3, 0.5, 0.7, 0.9\}$.

In the general case, there are several statistical strategies that one might adopt to construct a design having a given number of runs. One possibility is to select a simple random sample of points from the experimental region. In theory, there are infinitely many points between 0 and 1, and this makes selecting a simple random sample problematic. In practice, one only records numbers to a finite number of decimal places, and thus, the number of points between 0 and 1 can be regarded as finite. Therefore, one can assume the experimental region consists of finitely many points and select a simple random sample of these.

Simple random sampling in computer experiments can be quite useful. If the inputs are sampled according to some distribution (e.g., a distribution describing how the inputs are distributed in a given application), one can get a sense of how the corresponding outputs are distributed, and this can serve as the basis for inferences about the distribution of the output. However, for many purposes, other sampling schemes, such as stratified random sampling, are preferable to simple random sampling. Even if the goal is simply to guarantee that the inputs are evenly distributed over the experimental region, simple random sampling is not completely satisfactory, especially when the sample sizes are relatively small. With small samples in high-dimensional experimental regions, the sample will typically exhibit some clustering and fail to provide points in large portions of the region.

To improve the chances that inputs are spread "evenly" over the experimental region, one might use stratified random sampling. If a design consisting of n_s runs of the simulator is desired, one would divide the experimental region into n_s strata, spread evenly throughout the experimental region, and randomly select a single point from each. Varying the size and position of the strata, as well as sampling according to different distributions within the strata, allows considerable flexibility in selecting a design. This may be more or less useful, depending on the purpose of the computer experiment. For example, one may wish to explore some portions of the experimental region more thoroughly than others. However, if the goal is simply to select points that are spread evenly throughout the experimental region, spacing the strata evenly and sampling each according to a uniform distribution would seem the most natural choice.

If the output is thought to depend on only a few of the inputs (sometimes referred to as "factor sparsity"), then one might want to be sure that points are evenly spread across the projection of the experimental region onto these factors. A design that spreads points evenly throughout the full experimental region will not necessarily have this property. Alternatively, if one believes the model is well approximated by an additive model (a model that is the sum of terms that are each a function of only one of the inputs), a design that spreads points evenly across the range of each individual input (the one-dimensional projections) might be desirable. For n_s runs of the simulator, it can be difficult to guarantee that a design has such projection properties, even with stratified sampling. Latin hypercube sampling, the topic of the next subsection, is a way to generate designs that spread observations evenly over the range of each input separately.

5.2.2 Designs Generated by Latin Hypercube Sampling

Designs generated by Latin hypercube sampling are called Latin hypercube designs (LHD) throughout this book. Consider the simple case where the experimental region is the unit square $[0, 1]^2$. To obtain an LHD consisting of n_s points, divide each axis $[0, 1]$ into the n_s equally spaced intervals $[0, 1/n_s), \ldots, [(n_s - 1)/n_s, 1]$. This partitions the unit square into n_s^2 cells of equal size. Now, fill these cells with the integers $1, 2, \ldots, n_s$ so as to form a *Latin square*, i.e., by an arrangement in which each integer appears exactly once in each row and in each column of this grid of cells (see the left panel of Fig. 5.1). Select one of the integers at random. In each of the n_s cells containing this integer, select a point at random. The resulting n_s points are an LHD of size n_s; the right panel of Fig. 5.1 is an example of an LHD of run size $n_s = 3$ that can be constructed from the Latin square in the left panel.

Every LHD has points that are spread evenly over the values of each individual input variable. Of course, an LH sample could select points that are spread evenly along the diagonal of the square (see Fig. 5.3). Although the points in such a sample have projections that are evenly spread out over the values of each input variable separately, we would not regard them as evenly spread out over the entire unit square. Furthermore, recalling the discussion of space-filling in Sect. 5.2.1, LHDs consisting of n_s points along the diagonal do not become dense in the unit cube as n_s increases.

A general procedure for obtaining an LH sample of size n_s from $\boldsymbol{X} = (X_1, \ldots, X_d)$ when $X_1, \ldots, X_d$ are *independently distributed* will now be described. (Stein (1987) discusses the implementation of LH sampling when $\boldsymbol{X}$ has dependent components, but this case is not considered here.) First consider the case where each X_i component is uniformly distributed over $[0, 1]$ and an LH sample of size n_s is to be selected. Divide the $[0, 1]$ domain of each X_k, $1 \le k \le d$, into n_s intervals. The set of all possible Cartesian products of these intervals constitutes a partitioning of the d-dimensional sample space into n_s^d "cells." Select n_s cells from the n_s^d population of cells in such a way that the projections of the centers of each of the cells onto each of the d axes yield n_s distinct points; then choose a point at random in each selected cell.

Now consider the general case where X_k has marginal distribution $F_k(\cdot)$ and finite support $[a_k, b_k]$, $1 \le k \le d$. An LH sample over $\prod_{k=1}^{d}[a_k, b_k]$, is constructed as follows. Scale and shift each marginal random variable using $X_k = \frac{X_k - a_k}{b_k - a_k}$ so that now X_k can be assumed to have support $[0, 1]$; the inverse transform is used to place the support back on the original scale. Divide the kth axis into n_s parts, each of which has equal probability, $1/n_s$, under $F_k(\cdot)$; thus the division points for the kth axis are

$$F_k^{-1}\left(\frac{1}{n_s}\right), \ldots, F_k^{-1}\left(\frac{n_s - 1}{n_s}\right).$$

To choose n_s of the n_s^d cells so created, let $\boldsymbol{\Pi} = (\Pi_{jk})$ be an $n_s \times d$ matrix having permutations of $\{1, 2, \ldots, n_s\}$ as columns which are randomly selected from the set of all possible permutations. Then the "lower left-hand" coordinates of the jth cell in $\mathbb{R}^d$ are

$$F_k^{-1}\left(n_s^{-1}\left(\Pi_{jk} - 1\right)\right), \quad k = 1, \ldots, d, \quad j = 1, \ldots, n_s,$$

with the convention $F_k^{-1}(0) = 0$. For $j = 1, \ldots, n_s$, let X_{jk}, $k = 1, \ldots, d$, denote the kth component of the jth vector, $\boldsymbol{X}_j$. Then define the LH sample to have values

$$X_{jk} = F_k^{-1}\left(\frac{1}{n_s}\left(\Pi_{jk} - 1 + U_{jk}\right)\right),$$

where the $\{U_{jk}\}$ are independent and identically distributed $U[0, 1]$ deviates, for $j = 1, \ldots, n_s$ and $k = 1, \ldots, d$. In sum, the jth row of $\boldsymbol{\Pi}$ identifies the cell that $\boldsymbol{X}_j$ is sampled from, while the corresponding (independently generated) uniform deviates determine the location of $\boldsymbol{X}_j$ within the sampled cell. (The use of non-equal width probability intervals has been explored by Mease and Bingham (2006) and Dette and Pepelyshev (2010).)

Example 5.1. Suppose $\boldsymbol{X} = (X_1, X_2)$ is to be uniformly distributed over $[0, 1]^2$ so that $F_k^{-1}(w) = w$, $0 < w < 1$. To obtain an LH sample of size $n_s = 3$, say $\boldsymbol{X}_j = (X_{j1}, X_{j2})$, $j = 1, 2, 3$, compute

$$X_{jk} = F^{-1}\left(\frac{1}{3}\left(\Pi_{jk} - 1 + U_{jk}\right)\right) = \frac{1}{3}\left(\Pi_{jk} - 1 + U_{jk}\right)$$

for $j = 1,2,3$, $k = 1,2$, $\boldsymbol{\Pi} = (\Pi_{jk})$ is 3×2, and $\boldsymbol{U} = (U_{jk})$ has independent $U(0,1)$ components. The actual $\boldsymbol{X}$ sample depends on the randomly selected $\boldsymbol{\Pi}$ and the $\{U_{jk}\}_{j,k}$.

To envision the pattern of the LH sample, divide the unit interval in each dimension into [0,1/3), [1/3,2/3), and [2/3,1], yielding a partition of $[0, 1] \times [0, 1]$ into nine squares (cells) of equal area. In the LH sample, each of these subintervals will be represented exactly once *in each dimension*. For simplicity of discussion, suppose one labels these subintervals as 1, 2, and 3 in the order given above. One possible LHD would involve points randomly sampled from the (1,1), (2,3), and (3,2) squares and another possible design from the (1,2), (2,3), and (3,1) squares. Figure 5.1 plots the cells selected by the second design. These two selections correspond to the permutations

$$\boldsymbol{\Pi} = \begin{pmatrix} 1 & 1 \\ 2 & 3 \\ 3 & 2 \end{pmatrix} \text{ and } \boldsymbol{\Pi} = \begin{pmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 1 \end{pmatrix}. \tag{5.2.1}$$

Note that in each dimension, each subinterval appears exactly once. Because each subinterval is of length 1/3, the addition of $U_{jk}/3$ to the left-hand boundary of the selected subinterval serves merely to pick a specific point in it. ♦

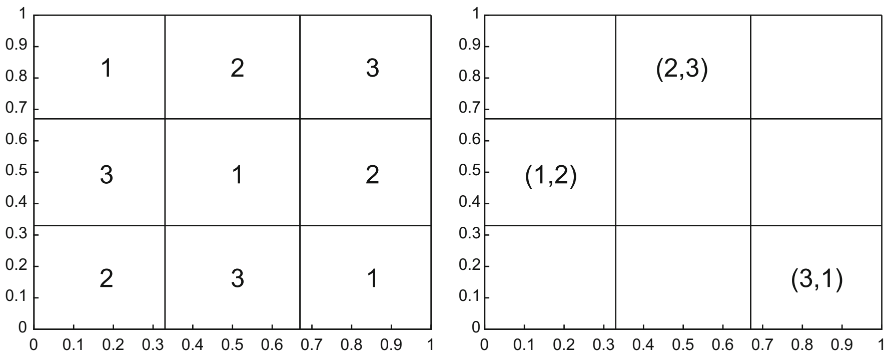
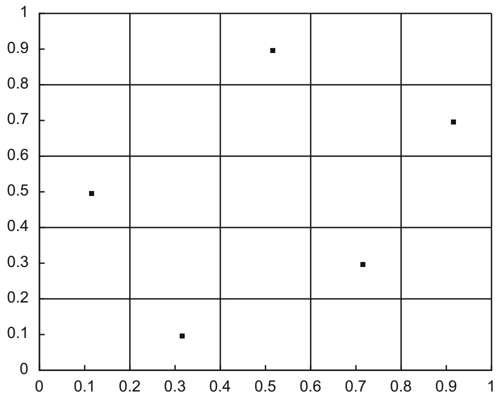


Fig. 5.1 Left panel: a 3×3 Latin square; right panel: the Latin hypercube sample (1,2), (2,3), and (3,1) which can be constructed from the Latin square

In the computer experiment setting, the input variables are not regarded as random for purposes of determining the experimental design and hence are denoted $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$. As in Example 5.1, suppose that each input variable has been scaled to have domain [0,1]. Denoting the k^{th} component of $\boldsymbol{x}_j$ by $x_{j,k}$ for $k = 1, \ldots, d$, suppose one obtains an LHD from a given $\boldsymbol{\Pi}$ as follows:

$$x_{j,k} = \frac{\Pi_{jk} - 0.5}{n_s}, \quad j = 1, \ldots, n_s;\ k = 1, \ldots, d\,.$$

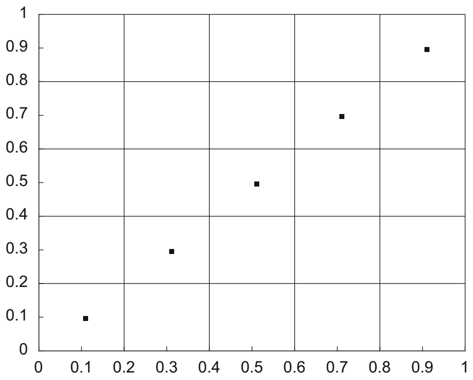
This corresponds to taking $U_{jk} = 0.5$ for each $j = 1, \ldots, n_s$ and $k = 1, \ldots, d$ rather than as a sample from a $U[0, 1]$ distribution. The "cells" are now identified with all d-dimensional Cartesian products of the intervals $\{[0, \frac{1}{n_s}), [\frac{1}{n_s}, \frac{2}{n_s}), \ldots, [1 - \frac{1}{n_s}, 1]\}$, and each $\boldsymbol{x}_j$ is sampled from the *center* of the cell indicated by the j^{th} row of $\boldsymbol{\Pi}$. An example of an LHD for $n_s = 5$ and $d = 2$ is given in Fig. 5.2 with its associated $\boldsymbol{\Pi}$ matrix.



$$\boldsymbol{\Pi} = \begin{pmatrix} 3 & 5 \\ 4 & 2 \\ 2 & 1 \\ 1 & 3 \\ 5 & 4 \end{pmatrix}$$

Fig. 5.2 A space-filling Latin hypercube design and the corresponding permutation $\boldsymbol{\Pi}$

As mentioned previously, LHDs need not be space-filling over the full experimental region. To illustrate this point visually, consider the LHD for $n_s = 5$ and $d = 2$ that is shown in Fig. 5.3, which one might *not* view as space-filling.



$$\boldsymbol{\Pi} = \begin{pmatrix} 1 & 1 \\ 2 & 2 \\ 3 & 3 \\ 4 & 4 \\ 5 & 5 \end{pmatrix}$$

Fig. 5.3 A non-space-filling Latin hypercube design and the corresponding permutation $\boldsymbol{\Pi}$

One consequence of computing responses at this set of inputs is that one would expect a predictor fitted using this design to generally perform well only for $x_1 \approx x_2$. For example, consider the deterministic function

$$y(x_1, x_2) = \frac{x_1}{1 + x_2}, \quad \mathcal{X} = [0, 1] \times [0, 1] .$$

The MLE-EBLUP (Sect. 3.3) was fitted to the observed responses using the training data for both of the designs shown in Figs. 5.2 and 5.3. The predictor was based on the stochastic process

$$Y(x_1, x_2) = \beta_0 + Z(x_1, x_2),$$

where $Z(\cdot)$ is a zero mean Gaussian process (GP) with unknown process variance and product power exponential correlation function (2.2.11).

The prediction error $|y(x_1, x_2) - \widehat{y}(x_1, x_2)|$ was calculated on a grid of 100 equally spaced (x_1, x_2) points for each design. Figure 5.4 plots a comparison of the prediction errors for the two designs where the symbol "1" ("0") indicates that the prediction error for the design of Fig. 5.3 is larger (smaller) than the prediction error for the design of Fig. 5.2. The space-filling design of Fig. 5.2 clearly yields a better predictor over most of the design space except near the diagonal where the LHD in Fig. 5.3 collects most of its data.

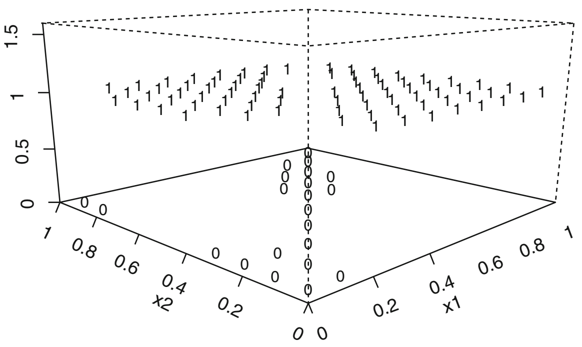


Fig. 5.4 Comparison of the LHDs in Figs. 5.2 and 5.3. The plotting symbol "1" ("0") at location (x_1, x_2) means that the LHD in Fig. 5.2 had lower (higher) mean squared prediction error than the LHD in Fig. 5.3

It is apparent from this discussion that although all LHDs possess desirable *marginal* properties, only a subset of these designs are truly "space-filling." Section 5.3 will discuss design criteria that have been successfully applied to select space-filling LHDs for use in computer experiments.

LHDs have been used extensively in the computer experiments literature; see, for example, Welch et al. (1992) and Bernardo et al. (1992). Other examples include Kennedy and O'Hagan (2001), Butler (2001), and Craig et al. (2001). Because of their widespread use, it is worth examining in some detail the properties of LHDs in the setting where all inputs are environmental variables.

Designs based on LH sampling were introduced by McKay et al. (1979) as a competitor to simple random sampling and stratified sampling when estimating the mean, variance, or distribution function of an output random variable. Stein (1987) and Owen (1992a) established additional large sample properties of LH sampling for estimating the mean $E[Y]$. Looking carefully at some of the results in these papers will provide greater insight into the actual properties of LHDs. It will then be worthwhile to reconsider their use in computer experiments.

5.2.3 Some Properties of Sampling-Based Designs

Suppose that a random vector of inputs $\boldsymbol{X} = (X_1, \ldots, X_d)$ to the computer output $y(\cdot)$ is distributed according to the known joint distribution $F(\cdot)$ over the experimental region $\mathcal{X} \equiv [0, 1]^d \subset \mathbb{R}^d$ (possibly after shifting and rescaling). Based on a sample $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_{n_s}$ from the distribution $F(\cdot)$, suppose one is interested in estimating the mean of $g(Y)$, assumed to exist and finite, where $g(\cdot)$ is a known function of the real-valued argument and $Y = y(\boldsymbol{X})$. This mean is

$$\mu = E[g(Y)] = \int_{\mathcal{X}} g(y(\boldsymbol{x}))\, dF(\boldsymbol{x}).$$

Now consider the properties of the naive moment estimator

$$T = T\left(y(\boldsymbol{X}_1), \ldots, y(\boldsymbol{X}_{n_s})\right) = \frac{1}{n_s} \sum_{j=1}^{n_s} g(y(\boldsymbol{X}_j))$$

when $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_{n_s}$ are either a simple random sample, a stratified random sample, or a Latin hypercube sample. To derive the properties of T, assume that the coordinates of $\boldsymbol{X}$ are independent, each with cumulative distribution function $F(\cdot)$. Let

$$\sigma^2 = Var[g(Y)]\,.$$

For clarity denote the estimator T by T_R when simple random sampling is used, by T_S when stratified random sampling is used, and by T_L when LH sampling is used. McKay et al. (1979) show the following:

Theorem 5.1. 1. If proportional sampling is used, i.e., if the sample size for stratum i is proportional to the probability under $F(\cdot)$ of a point belonging to stratum i, then $Var[T_S] \le Var[T_R]$. 2. If $y(x_1, \ldots, x_d)$ is monotonic in each of its arguments and $g(w)$ is a monotonic function of $w \in \mathbb{R}$, then $Var[T_L] \le Var[T_R]$.

Section 5.7.1 of the Chapter Notes provides a proof of the second part of this theorem.

At this point a few cautions are in order. First, these results show only that for estimating the expected value of $g(Y)$ over the experimental region, designs based on proportional sampling are better than those based on simple random sampling, and, under certain conditions, LHDs are better than those based on simple random sampling. Designs based on LH sampling need not always be better than designs based on simple random sampling nor is it known whether designs based on LH sampling are better than other types of designs, such as stratified sampling. Note, however, that the formulas derived in McKay et al. (1979) do allow one to compare designs based on LH and stratified proportional sampling.

Second, in most computer experiments, one does not know the relationship between the output $y(\boldsymbol{x})$ and the component inputs $x_1, \ldots, x_d$. It is unlikely that one would be willing to assume this relationship is monotonic. And if one makes such an assumption, the conditions on $g(\cdot)$ given in Theorem 5.1 imply that the extrema of $g(\cdot)$ are on the boundary of the experimental region. If, as is often the case, one is interested in finding the extrema of $g(\cdot)$ and one knows the extrema are on the boundary of the experimental region, one would want to take observations near or on the boundary rather than using an LHD.

Third, the above properties are relevant if one is interested in estimating the expected value of $g(Y)$ over the experimental region. To illustrate, let $I\{E\}$ denote the indicator function (1 or 0, as E is true or false) and y_{fixed} be a given point in $\mathbb{R}$. Then setting $g(y) = y$ yields the mean of Y over the experimental region, while setting $g(y) = I\{y \leq y_{fixed}\}$ produces the cumulative distribution function of Y at y_{fixed}. However, finding the expected value of $g(Y)$ over the experimental region is not usually the goal in computer experiments. More typically, the goal is to fit a model that approximates $g(\cdot)$ over the experimental region or to determine the points in the experimental region that are extrema of $g(\cdot)$. Thus, although LHDs are quite popular in computer experiments, the above results do not indicate whether they have good properties in many of the situations where computer experiments are conducted. Better justification for the use of LHDs comes from the results to be discussed next.

Additional properties of sample means based on Latin hypercube samples have been established by Stein (1987) and Owen (1992a). The *remainder of this section* takes $g(y) = y$ and uses $\overline{Y} = \frac{1}{n_s} \sum_{j=1}^{n_s} y(\boldsymbol{X}_j)$ to estimate $E[y(\boldsymbol{X})]$ where, recall, $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_{n_s}$ is a random sample with distribution $\boldsymbol{X}$. Let $F_i(\cdot)$ denote the marginal distribution of X_i, the i^{th} coordinate of $\boldsymbol{X}$. As above, assume the coordinates of $\boldsymbol{X}$ are independent so

$$F(\boldsymbol{x}) = \prod_{i=1}^{d} F_i(x_i).$$

For $1 \leq j \leq d$, let $\boldsymbol{X}_{-j}$ denote $\boldsymbol{X}$ omitting X_j,

$$F_{-j}(\boldsymbol{x}_{-j}) = \prod_{i=1, i \neq j}^{d} F_i(x_i)$$

the distribution function of $\boldsymbol{X}_{-j}$, $\boldsymbol{x}_{-j}$ the corresponding argument extracted from $\boldsymbol{x}$, and $\mathcal{X}_{-j}$ denote the support of $F_{-j}(\cdot)$. Assuming $\int_{\mathcal{X}} y^2(\boldsymbol{x})\, dF(\boldsymbol{x}) < \infty$, decompose $y(\boldsymbol{x})$ as follows. Define

$$\mu = \int_{\mathcal{X}} y(\boldsymbol{x})\, dF(\boldsymbol{x}) \quad \text{and} \quad \alpha_j(x_j) = \int_{\mathcal{X}_{-j}} (y(\boldsymbol{x}) - \mu)\, dF_{-j}(\boldsymbol{x}_{-j})\,.$$

Then μ is the overall mean, the $\{\alpha_j(x_j)\}$ are the "main effect" functions corresponding to the coordinates of $\boldsymbol{x}$, and $r(\boldsymbol{x}) = y(\boldsymbol{x}) - \mu - \sum_{i=1}^{d} \alpha_i(x_i)$ is the residual (from additivity) of $y(\boldsymbol{x})$. These quantities are continuous analogs of an "analysis of variance" decomposition of $y(\boldsymbol{x})$. Further reason for this designation is the fact that

$$\int_0^1 \alpha_j(x_j)\, dF_j(x_j) = 0 \quad \text{and} \quad \int_{\mathcal{X}_{-j}} r(\boldsymbol{x})\, dF_{-j}(\boldsymbol{x}_{-j}) = 0$$

for any x_j and all j (see also Sect. 7.4).

Stein (1987) shows that for large samples, $Var[\overline{Y}]$ is smaller under LH sampling than simple random sampling unless all main effect functions are 0. To be precise, Stein (1987) proves the following expansions for the variance of $\overline{Y}$ under the two sampling schemes:

Theorem 5.2. Under Latin hypercube sampling and simple random sampling we have

$$Var_{LHS}\left[\overline{Y}\right] = \frac{1}{n_s}\int_{\mathcal{X}} r^2(\boldsymbol{x})\, dF(\boldsymbol{x}) + o(n_s^{-1}) \quad \text{and}$$

$$Var_{SRS}\left[\overline{Y}\right] = \frac{1}{n_s}\int_{\mathcal{X}} r^2(\boldsymbol{x})\, dF(\boldsymbol{x}) + \frac{1}{n_s}\sum_{i=1}^{d}\int_{a_i}^{b_i} \alpha_i^2(x_i)\, dF_i(x_i) + o(n_s^{-1}),$$

respectively.

The implication of this expansion is that, unless all $\alpha_j(\cdot)$ are identically 0, in the limit LH sampling has a smaller (order $1/n_s$) variance than simple random sampling.

Further, not only can the variance of $\overline{Y}$ be estimated but also the normality of $\overline{Y}$ can be established. For simplicity, assume $\mathcal{X} = [0,1]^d$ and that $F(\cdot)$ is uniform. More general cases can often be reduced to this setting by appropriate transformations. Owen (1992a) shows that $\overline{Y}$ computed from inputs based on LH sampling is approximately normally distributed for large samples under mild conditions; he proves:

Theorem 5.3. If $y(\boldsymbol{x})$ is bounded, then under LH sampling, $\sqrt{n_s}\,(\overline{Y} - \mu)$ tends in distribution to $N\left(0, \int_{\mathcal{X}} r^2(\boldsymbol{x})\, d\boldsymbol{x}\right)$ as $n_s \to \infty$.

Theorem 5.3 can be used as the basis for statistical inference about μ. Owen (1992a) also provides estimators of the asymptotic variance

$$\int_{\mathcal{X}} r^2(\boldsymbol{x})\, d\boldsymbol{x}$$

to facilitate application of these results to computer experiments.

Section 5.7.2 of the Chapter Notes describes the use of LHDs in a generalization of these constant mean results to a regression setting, which has potential for use in computer experiments.

5.3 Latin Hypercube Designs with Additional Properties

Figure 5.3 displays an LHD that would probably not be considered space-filling in $[0, 1]^2$ because the points lie along a straight line and are perfectly correlated. By comparison, the LHD in Fig. 5.2 appears more space-filling in $[0, 1]^2$, and the points appear much less correlated. Is it possible to identify special types of LHDs, or extensions of LHDs, that have desirable properties? For example, are there LHDs that are space-filling in $\mathcal{X}$ or that have space-filling projections onto subspaces of dimension greater than 1? As in the previous section, assume (after possible rescaling) that $\mathcal{X} = [0, 1]^d$.

5.3.1 Latin Hypercube Designs Whose Projections Are Space-Filling

One extension of LHDs, based on what are known as orthogonal arrays, produces designs with attractive projection properties. An $n_s \times d$ matrix $\boldsymbol{O}$ all of whose entries come from a given set of s symbols are said to be an *orthogonal array* (OA) of *strength* t ($t \leq d$) provided that in every $n_s \times t$ submatrix of $\boldsymbol{O}$, all s^t possible rows appear the same number of times. The collection of all $n_s \times d$ OAs having entries from a set of s symbols of strength t is denoted $OA(n_s, d, s, t)$. If $\boldsymbol{O} \in OA(n_s, d, s, t)$, the common number of times that all s^t possible rows appear in any $n_s \times t$ submatrix is denoted by λ. For detailed discussions of the properties, applications, and construction of orthogonal arrays, see Raghavarao (1971), Hedayat et al. (1999), or Wu and Hamada (2009). A large library of OAs is available at the website http://neilsloane.com/oadir/ maintained by Neal Sloane.

Example 5.2. The simplest OA has strength $t = 1$. For example, any $s \times d$ matrix each of whose columns are the integers $1, 2, \ldots, s$ in some order is an $OA(s, d, s, 1)$ (with $\lambda = 1$). For example, if $s = 5$ and $d = 2$,

$$\begin{pmatrix} 3 & 5 \\ 4 & 2 \\ 2 & 1 \\ 1 & 3 \\ 5 & 4 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1 & 1 \\ 2 & 2 \\ 3 & 3 \\ 4 & 4 \\ 5 & 5 \end{pmatrix}$$

are both $OA(5, 2, 5, 1)$ arrays. Orthogonal arrays of strength $t > 1$ are more challenging to construct. It is easy to check that

$$\boldsymbol{O} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 3 & 3 \\ 2 & 1 & 2 \\ 2 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 3 \\ 3 & 2 & 1 \\ 3 & 3 & 2 \end{pmatrix} \tag{5.3.1}$$

is an $OA(9, 3, 3, 2)$, again with $\lambda = 1$. Geometrically, the same nine points result from projecting the 3-D points of the OA onto any 2-D subspace. ♦

Every OA having strength $t = 1$ and $\lambda = 1$ can be transformed into an LHD as follows. Continuing Example 5.2, suppose that $\boldsymbol{O} = (O_{j,k}) \in OA(s, d, s, 1)$ which necessarily has $\lambda = 1$. Then let $\boldsymbol{X}$ be the $s \times d$ matrix with entry

$$x_{j,k} = \frac{O_{j,k} - 0.5}{s}$$

in row j and column k, $j = 1, \ldots, s$ and $k = 1, \ldots, d$. The s rows of $\boldsymbol{X}$ determine s points in $[0, 1]^d$. By construction, $\boldsymbol{X}$ is an s-point LHD in $[0, 1]^d$ (see the discussion following (5.2.1)). Notice that the two $OA(5, 2, 5, 1)$ designs in Example 5.2 produce the LHDs in Figs. 5.2 and 5.3.

Because of this connection between orthogonal arrays having $t = 1 = \lambda$ and LHDs, it will not be completely surprising to learn that orthogonal arrays of strength $t \geq 2$ can be used to define extensions of LHDs. In particular, orthogonal arrays of strength $t \geq 2$ can be used to generate designs with the property that all projections of the points in the design onto any t (or fewer) dimensions are "space-filling." Owen (1992b) and Tang (1993) describe two approaches to use orthogonal arrays to form LHDs with higher dimensional, space-filling projections.

The Owen (1992b) procedure for generating an n_s-run space-filling design in $[0, 1]^d$ from the columns of an $n_s \times d$ orthogonal array is as follows. Suppose that the generating orthogonal array has strength t, then projections of the resulting design in t or fewer of the coordinates will form a regular grid. Example 5.3 illustrates the method in $d = 3$ dimensions starting with an orthogonal array of strength $t = 2$.

Example 5.3. Start with the 9×3 OA $\boldsymbol{O}$ in (5.3.1) which has $s = 3$ symbols and strength $t = 2$. To construct a design on the unit cube $[0, 1]^3$ from $\boldsymbol{O}$, divide the cube into a 3×3×3 grid of 27 equi-volume cells. Label the cells using (1,1,1) to denote the cell $[0, \frac{1}{3}] \times [0, \frac{1}{3}] \times [0, \frac{1}{3}]$, (1,1,2) to denote the cell $[0, \frac{1}{3}] \times [0, \frac{1}{3}] \times [\frac{1}{3}, \frac{2}{3}]$, (1,1,3) to denote the cell $[0, \frac{1}{3}] \times [0, \frac{1}{3}] \times [\frac{2}{3}, 1], \ldots,$ and (3,3,3) the cell $[\frac{2}{3}, 1] \times [\frac{2}{3}, 1] \times [\frac{2}{3}, 1]$. Each row of $\boldsymbol{O}$ corresponds to one of these 27 cells. The points in the centers of the nine cells form a nine-run design on $[0, 1]^3$. Projected onto any 2-D subspace, the design forms a regular 3×3 grid. Projected onto any 1-D subspace (axis), the design produces a regular grid at the centers of the intervals $[0, \frac{1}{3}]$, $[\frac{1}{3}, \frac{2}{3}]$, and $[\frac{2}{3}, 1]$; there are three runs projecting onto each grid point (hence the 1-D projections are not space-filling in the way in which LHDs are). Selecting a point at random from each of these cells results in a design whose projections onto any two-dimensional subspace, while not a regular grid, is space-filling within the nine blocks of the subspace. This random selection of points would prevent the 1-D projections from yielding multiple points at the same locations. ♦

Alternatively, Tang (1993) uses an OA to specify a restricted permutation of $\{1, 2, \ldots, n_s\}$ so that t and lower dimensional subspaces have similar space-filling projections to those of Owen (1992b). Fix $\boldsymbol{O} \in OA(n_s, d, s, t)$. A permutation $\boldsymbol{\Pi}$ of the integers $1, \ldots, n_s$ is selected for each column of $\boldsymbol{O}$ as follows. Suppose that λ is the number of times that each of the s^t rows appears in any $n_s \times t$ subarray of $\boldsymbol{O}$. Then each $k \in \{1, \ldots, s\}$ appears λs^{t-1} times in every column of $\boldsymbol{O}$. Fix a column, and replace the λs^{t-1} occurrences of k by a permutation of the integers $(k-1)\lambda s^{t-1} + 1, (k-1)\lambda s^{t-1} + 2, \ldots, k\lambda s^{t-1}$. The LHD formed from this $\boldsymbol{\Pi}$ will have projections onto all subspaces of dimension $\leq t$ uniformly distributed (including, of course, univariate projections). Tang (1993) refers to an LHD constructed in this way as an OA-based LHD. Note that in the step for constructing $\boldsymbol{\Pi}$ from the initial orthogonal array, many choices are possible for the permutations; hence from a given initial orthogonal array, many OA-based LHDs can be constructed. One can impose an additional desirable criterion to select among the resulting LHDs.

Example 5.4. Start with the $OA(4, 2, 2, 2)$

$$\boldsymbol{O} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 2 & 1 \\ 2 & 2 \end{pmatrix}.$$

To obtain $\boldsymbol{\Pi}$, in each column of $\boldsymbol{O}$, replace the symbol 1 with a random permutation of the integers $\{1, 2\}$ and the symbol 2 with a random permutation of the integers $\{3, 4\}$. For example, suppose that in column 1, the first occurrence of the symbol 1 is replaced by 1 and the second occurrence of the symbol 1 is replaced by 2. Also replace the first occurrence of the symbol 2 in column 1 by 4 and the second occurrence of the symbol 2 by 3. Then in column 2 replace the first occurrence of the symbol 1 by 2 and the second occurrence of the symbol 1 by 1, and replace the first occurrence of the symbol 2 in column 2 by 3 and the second occurrence of the symbol 2 by 4. This gives the LHD

$$\begin{pmatrix} 1 & 2 \\ 2 & 3 \\ 4 & 1 \\ 3 & 4 \end{pmatrix}$$

♦

Example 5.5. Again start with the 9×3 OA $\boldsymbol{O}$ in (5.3.1). To obtain $\boldsymbol{\Pi}$, in each column replace the symbol 1 with a random permutation of the integers $\{1, 2, 3\}$, the symbol

2 with a random permutation of the integers {4, 5, 6}, and the symbol 3 by a random permutation of the integers {7, 8, 9}. As illustrated in Example 5.4, the permutations can be different in each column. For example, use (1, 3, 2), (3, 2, 1), and (2, 1, 3) for the 1's in columns 1, 2, and 3, respectively. Use (6, 5, 4), (4, 5, 6), and (5, 4, 6) for the 2's in columns 1, 2, and 3, respectively. Finally use (9, 7, 8), (8, 7, 9), and (7, 9, 8) for the 3's in columns 1, 2, and 3, respectively. This gives the LHD

$$\begin{pmatrix} 1 & 3 & 2 \\ 3 & 4 & 5 \\ 2 & 8 & 7 \\ 6 & 2 & 4 \\ 5 & 5 & 9 \\ 4 & 7 & 1 \\ 9 & 1 & 8 \\ 7 & 6 & 3 \\ 8 & 9 & 6 \end{pmatrix}. \tag{5.3.2}$$

Selecting a random point from each of the cells of the (scaled) version of the LHD (5.3.2) gives the three 2-D projections shown in Fig. 5.5. These plots show that 2-D projections of the OA-based LHD are space-filling in the sense of placing one run in each of the nine equal-sized square cells of $[0, 1]^2$. ♦

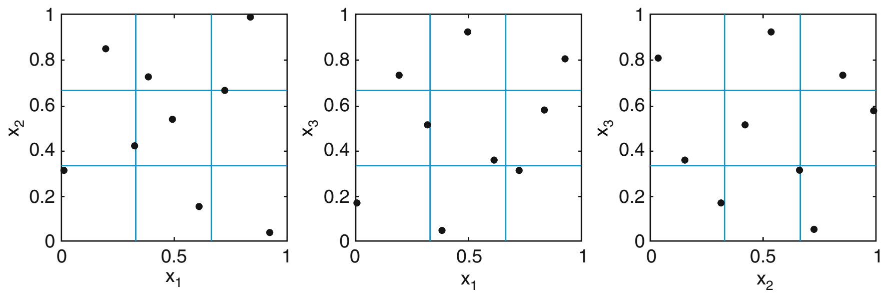


Fig. 5.5 The three 2-D projections of a scaled and randomized version of the OA-based LHD given in (5.3.2)

Another way to think about an OA-based LHD is that the structure of the orthogonal array is used to restrict the placement of the points within the unit hypercube (for this discussion assume that our interest is in LHDs on the d-dimensional unit hypercube). In the context of the previous discussion, for the kth column of $\boldsymbol{O} \in OA(n_s, d, s, t)$, consider the non-overlapping division of $[0, 1]$ into s equal length intervals of the form $[0, \frac{1}{s}) \cup [\frac{1}{s}, \frac{2}{s}) \cup \cdots \cup [\frac{s-1}{s}, 1]$. Because $\boldsymbol{O}$ is an orthogonal array, each of the s symbols appears equally often in each column; let r denote the number of times each symbol appears in a given column. For a given level $l_j = 0, 1, \ldots, s-1$ define the non-overlapping division of the interval $[\frac{l_j}{s}, \frac{l_j+1}{s})$ into the r subintervals

$$\left[\frac{l_j}{s} + \frac{i}{sr}, \frac{l_j}{s} + \frac{i+1}{sr}\right), \quad i = 0, 1, \ldots, r-1.$$

For column k let $p_{k_1}, p_{k_2}, \ldots, p_{k_r}$, be a random permutation of the integers $0, 1, \ldots, r-1$. Then the r points corresponding to level l_j are randomly (or systematically) placed one each in the Cartesian product of intervals

$$\left[\frac{l_j}{s} + \frac{p_{k_i}}{sr}, \frac{l_j}{s} + \frac{p_{k_i}+1}{sr}\right), \quad k = 1, 2, \ldots, d.$$

Notice for each column of $OA(n_s, d, s, t)$ that $n_s = rs$ and the Latin hypercube intervals $[\frac{i}{n_s}, \frac{i+1}{n_s})$ are identical to the substratification described so that the resulting array, with placement of points imposed by the strength t orthogonal array, is indeed an LHD with t-dimensional projection properties consistent with $OA(n_s, d, s, t)$.

Although the methods above extend the projection properties of LHDs to higher dimensions, they have the drawback that the base OAs exist only for certain values of n_s and d. Indeed, because n_s must be be a constant multiple of s^t, they need not be available for the larger s and t that would be desirable in some computer experiment applications. Recently Loeppky et al. (2012) introduced a more flexible class of designs, called projection array-based designs that have space-filling properties analogous to OA-based LHDs but which exist for all run sizes.

Another criterion for selecting space-filling LHDs is based on examining the correlations of the input variables. The idea for this approach is the observation that scatterplots of uncorrelated variables "appear" more space-filling than do plots of highly correlated variables. This suggests that a possible strategy for avoiding LHDs that do not appear to be space-filling in $[0, 1]^d$ is to select those for which the points are uncorrelated. The Owen (1992b) and Tang (1993) methods for constructing LHDs with an underlying orthogonal array structure can be viewed as assuring that, in some dimensions, the points in the LHD appear uncorrelated. Section 5.3.3 gives a fuller discussion of orthogonal LHDs. Finally, Cioppa and Lucas (2007) explore methods for forming nearly orthogonal designs, not necessarily LHDs, for arbitrary numbers of runs.

5.3.2 Cascading, Nested, and Sliced Latin Hypercube Designs

Cascading LHDs, nested LHDs, and sliced LHDs are LHDs with additional properties. Introduced in Handcock (1991), *cascading LHDs* allow one to explore both the local (in small subregions) and the global (over the entire experimental region) behavior of the response. Cascading LHDs can be described as follows. Generate an LHD. At each point of this design, consider a small region around the point. In this small region, generate a second LHD. The result is a cluster of small LHDs and is called a cascading Latin hypercube design.

Qian (2009) introduced *nested LHDs*. A nested LHD having n_s runs and a layers is an LHD with the property that it can be used to generate a series of $a-1$

successively smaller LHDs having fewer runs. One application of nested LHDs is to computer experiments involving codes with multiple levels of accuracy, in which experiments with higher levels of accuracy are more expensive (and hence are observed at fewer points) than those of lower accuracy. An n_s-run nested LHD can be used to determine the points at which the lowest accuracy experiment is run. The successively smaller layers of this LHD can be used to determine the points at which the successively higher accuracy experiments are run. Using a nested LHD guarantees that the points at which any particular level-of-accuracy experiment is run are also observed at all lower accuracy experiments. See Kennedy and O'Hagan (2000), Qian et al. (2006), and Qian and Wu (2008) for more on such experiments.

Nested LHDs with a given number of run sizes need not exist. For example, suppose one uses an LHD consisting of n_s runs in $\mathbb{R}^d$. After fitting a predictor to the data, suppose one decides the fit is inadequate and wishes to make m_s additional runs of the computer simulator. Is it possible to select the m_s runs in such a way that the resulting set of $n_s + m_s$ runs is an LHD? The answer is no. Figure 5.6 displays a two-point LHD in two dimensions with the two points randomly placed in two of the four cells (outlined by the solid lines). This cannot be extended to a three-point LHD in two dimensions, because both points are in the same cell when the design space is partitioned into nine cells (outlined by the dashed lines). However, the two-point LHD could be extended to a four-point LHD in two dimensions because the two points would now be in two separate cells when the design space is partitioned into 16 cells.

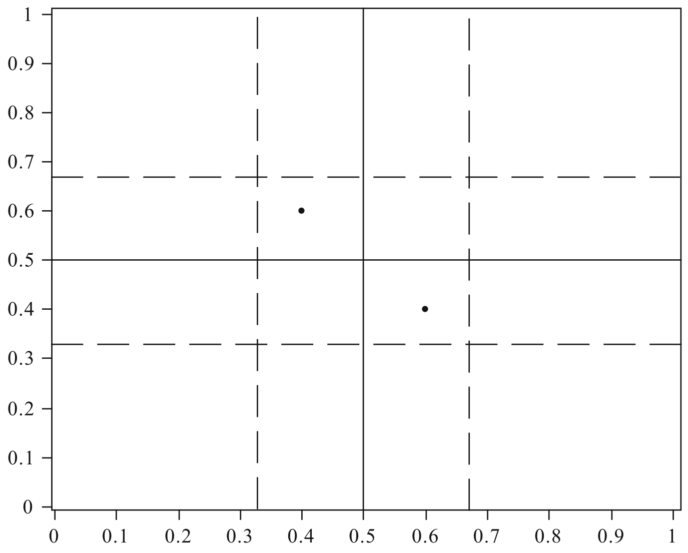


Fig. 5.6 A two-point LHD that cannot be extended to a three-point LHD. Points are placed at random in the four cells for a two-point LHD. The cells are outlined by the solid lines. The dashed lines outline the nine cells for a three-point LHD. Notice both points are in the same cell

Returning to the case of the previous paragraph, one might ask when will a nested LHD having n_s and m_s runs exist? Suppose the runs of the original LHD were chosen at random in the n_s cells and $m_s = an_s$ for some positive integer a. Then it is possible to add the m_s points in such a way the $m_s + n_s$ points form an LHD. In the initial LHD, the domain of each input variable must be subdivided into n_s intervals. Subdivide each of these n_s intervals into $a + 1$ intervals so that now the domain of each input variable is subdivided into $(a + 1)n_s$ intervals. The Cartesian product of these intervals constitutes a partitioning of the d-dimensional input space into $[(a + 1)n_s]^d$ cells. Each of the n_s points in the original design is in exactly one of these cells. Choose a subset of $(a + 1)n_s$ cells in such a way that they include the n_s cells containing points from the original design *and* so that the projections of the centers of all $(a + 1)n_s$ points onto each component axis yield $(a + 1)n_s$ distinct points on the axis. In terms of the method described before Example 5.1, this will mean that one can select only certain $(a + 1)n_s \times d$ matrices $\boldsymbol{\Pi}$ having permutations of $\{1, 2, \ldots, (a+1)n_s\}$ as columns. Notice that if in the original LHD the points were chosen at the center of the n_s cells, it is still possible to add m_s points in such a way that the resulting design is an LHD with points at the center of cells, provided a is even.

Qian (2009) also investigated the properties of nested LHDs for estimating the means of functions and showed that nested LHDs can outperform i.i.d. sampling under conditions analogous to those in Theorem 5.1.

Instead of adding points to an existing LHD in such a way that the result is also an LHD, one could consider the "reverse" problem. Do there exist LHDs with the property that they can be divided into several smaller LHDs? One possibility is to generate an an_s-point LHD as described in the previous paragraphs. First generate an n_s-point LHD. Add $(a - 1)n_s$ points as described in the previous paragraphs to generate an an_s-point LHD. By construction, the resulting design is both an LHD and contains a subset of n_s points (the starting design) which is also an LHD. More generally, one could start with an n_s-point LHD, extend it to an a_1n_s-point LHD, then extend this a_1n_s-point LHD to an $a_1a_2n_s$-point LHD, and continue on to an $a_1a_2 \cdots a_bn_s$-point LHD. The final design contains subsets of points that are n_s-point, a_1n_s-point, $a_1a_2n_s$-point, ..., , and $a_1a_2 \cdots a_{b-1}n_s$-point LHDs.

Qian (2012) introduced another type of LHD which he called a *sliced LHD*. A sliced LHD for d inputs with $n_s = am_s$ runs has the property that the LHD can be subdivided into a LHDs of size $m_s \times d$. These a divisions of the n_s-run LHD are called slices and the original n_s-run LHD a sliced LHD. The motivation for such a design results from considering a computer experiment having one or more quantitative inputs and a single qualitative input having a values. Each slice provides a space-filling design for m_s runs of the code at a fixed value of the qualitative variable. If the qualitative variable has no significant effect on the response, the slices collapse into a larger LHD.

Qian (2012) showed that sliced LHDs can outperform both i.i.d. sampling and standard LH sampling in terms of variance reduction in settings where one wishes to estimate a weighted average of expected values of a functions under conditions analogous to those in Theorem 5.1.

Two recent extensions of LHDs can be found in Ba and Joseph (2011) and Joseph et al. (2015). Ba and Joseph (2011) discussed a class of designs, called multilayer designs, that have good space-filling properties. These designs are an alternative to LHDs and are developed by splitting two-level factorial designs into multiple layers. Joseph et al. (2015) introduced a maximum projection criterion that produces LHDs with good projection properties (see Sect. 5.4).

5.3.3 Orthogonal Latin Hypercube Designs

Another attempt to find LHDs that have additional good properties is due to Ye (1998). He discussed a method for constructing LHDs for which all pairs of columns are *orthogonal* to each other, where a pair of columns is defined to be orthogonal if their inner product is zero. Recall that the columns of an $n_s \times d$ LHD are formed from an $n_s \times d$ matrix $\boldsymbol{\Pi}$ whose columns are permutations of the integers $\{1, 2, \ldots, n_s\}$. By restricting to only certain permutations of $\{1, 2, \ldots, n_s\}$, Ye (1998) was able to generate LHDs with columns, once properly reflected and rescaled, that are orthogonal; he terms these orthogonal Latin hypercubes (OLHs). Ye's method of construction for $n_s = 2^{m-1}$ and $d = 2m - 2$ where m is an integer > 1 is as follows.

Let $\boldsymbol{e}$ be the $2^{m-1} \times 1$ column vector with entries $\{1, 2, \ldots, 2^{m-1}\}$. Ye (1998) uses the notation $(r \quad s)$ to represent the permutation of rows of $\boldsymbol{e}$ (more generally, the permutation of the rows of any column vector) obtained by transposing rows r and s. For example, if $m = 3$, then $\boldsymbol{e} = (1, 2, 3, 4)^\top$ and $(1 \quad 3)$ of $\boldsymbol{e}$ is $(3, 2, 1, 4)^\top$. Products of such permutations denote the permutation resulting from applying each in turn, i.e., applying their composition. For example, $(2 \quad 4)(1 \quad 3)$ of $\boldsymbol{e} = (1, 2, 3, 4)^\top$ is $(2 \quad 4)$ of $(3, 2, 1, 4)^\top$, namely, $(3, 4, 1, 2)^\top$.

For a given integer $m > 1$, Ye (1998) defines the $m - 1$ permutations

$$A_k = \prod_{j=1}^{2^{m-k-1}} \left\{ \prod_{i=1}^{2^{k-1}} \Big((j-1)2^k + i \quad j2^k + 1 - i \Big) \right\}, k = 1, \ldots, m-1$$

where $\prod$ represents the product (or composition) of permutations. For example, if $m = 3$,

$$\begin{aligned} A_1 &= \prod_{j=1}^{2} ((j-1)2 + 1 \quad j2) \\ &= (1 \quad 2)(3 \quad 4) \end{aligned}$$

$$\begin{aligned} A_2 &= \prod_{i=1}^{2} (i \quad 4 + 1 - i) \\ &= (1 \quad 4)(2 \quad 3)\,. \end{aligned}$$

Next, let $\boldsymbol{M}$ be the $2^{m-1} \times (2m-2)$ matrix with columns

$$\left[\boldsymbol{e}\ \boldsymbol{A}_1\boldsymbol{e} \cdots \boldsymbol{A}_{m-1}\boldsymbol{e}\ \boldsymbol{A}_{m-1}\boldsymbol{A}_1\boldsymbol{e} \cdots \boldsymbol{A}_{m-1}\boldsymbol{A}_{m-2}\boldsymbol{e}\right].$$

For example, if $m = 3$, $\boldsymbol{M}$ is the 4×4 matrix

$$\boldsymbol{M} = \begin{bmatrix} 1 & 2 & 4 & 3 \\ 2 & 1 & 3 & 4 \\ 3 & 4 & 2 & 1 \\ 4 & 3 & 1 & 2 \end{bmatrix}.$$

For a given integer k between 1 and $m-1$, let

$$\boldsymbol{B}_{m-k} = \begin{pmatrix} -1 \\ 1 \end{pmatrix},\ \boldsymbol{B}_i = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

for all $i \neq m-k$ and define

$$\boldsymbol{a}_k = \bigotimes_{j=1}^{m-1} \boldsymbol{B}_j$$

where $\otimes$ is the Kronecker product. For example, if $m = 3$, with $k = 1$,

$$\begin{aligned} \boldsymbol{a}_1 &= \begin{pmatrix} 1 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} -1 \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} -1 \\ 1 \\ -1 \\ 1 \end{pmatrix} \end{aligned}$$

and with $k = 2$,

$$\begin{aligned} \boldsymbol{a}_2 &= \begin{pmatrix} -1 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} -1 \\ -1 \\ 1 \\ 1 \end{pmatrix}. \end{aligned}$$

Use $\odot$ to denote the elementwise product of two vectors, for example,

$$\begin{pmatrix} -1 \\ 1 \\ -1 \\ 1 \end{pmatrix} \odot \begin{pmatrix} -1 \\ -1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \end{pmatrix}.$$

Next, denote the $2^{m-1} \times 1$ vector of 1s by $\mathbf{1}$. Ye (1998) defined $\boldsymbol{S}$ to be the $2^{m-1} \times (2m-2)$ matrix with columns

$$\left[\mathbf{1}\ \boldsymbol{a}_1\ \cdots\ \boldsymbol{a}_{m-1}\ \boldsymbol{a}_1 \odot \boldsymbol{a}_2\ \cdots\ \boldsymbol{a}_1 \odot \boldsymbol{a}_{m-1}\right].$$

For example, when $m = 3$,

$$\boldsymbol{S} = \left[\mathbf{1}\ \boldsymbol{a}_1\ \boldsymbol{a}_2\ \boldsymbol{a}_1 \odot \boldsymbol{a}_2\right] = \begin{pmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \end{pmatrix}.$$

Let $\boldsymbol{T} = \boldsymbol{M} \odot \boldsymbol{S}$. For example, when $m = 3$,

$$\boldsymbol{T} = \boldsymbol{M} \odot \boldsymbol{S} = \begin{pmatrix} 1 & -2 & -4 & 3 \\ 2 & 1 & -3 & -4 \\ 3 & -4 & 2 & -1 \\ 4 & 3 & 1 & 2 \end{pmatrix}.$$

Consider the $(2^m + 1) \times (2m - 2)$ matrix $\boldsymbol{O}$ whose first 2^{m-1} rows are $\boldsymbol{T}$, whose next row consists of all 0s, and whose last 2^{m-1} rows are the "mirror image" of $\boldsymbol{T}$, namely, the rows of $-\boldsymbol{T}$ in reverse order. For example, when $m = 3$,

$$\boldsymbol{O} = \begin{pmatrix} 1 & -2 & -4 & 3 \\ 2 & 1 & -3 & -4 \\ 3 & -4 & 2 & -1 \\ 4 & 3 & 1 & 2 \\ 0 & 0 & 0 & 0 \\ -4 & -3 & -1 & -2 \\ -3 & 4 & -2 & 1 \\ -2 & -1 & 3 & 4 \\ -1 & 2 & 4 & -3 \end{pmatrix}.$$

From $\boldsymbol{O}$ remove the row consisting of all 0s and rescale levels to be equidistant. Let $\boldsymbol{O}^*$ denote the resulting $2^m \times (2m - 2)$ matrix. For example, when $m = 3$,

$$\boldsymbol{O}^* = \begin{pmatrix} 0.5 & -1.5 & -3.5 & 2.5 \\ 1.5 & 0.5 & -2.5 & -3.5 \\ 2.5 & -3.5 & 1.5 & -0.5 \\ 3.5 & 2.5 & 0.5 & 1.5 \\ -3.5 & -2.5 & -0.5 & -1.5 \\ -2.5 & 3.5 & -1.5 & 0.5 \\ -1.5 & -0.5 & 2.5 & 3.5 \\ -0.5 & 1.5 & 3.5 & -2.5 \end{pmatrix}.$$

Ye (1998) showed that the columns of $\boldsymbol{O}$ are orthogonal to each other, the elementwise square of each column of $\boldsymbol{O}$ is orthogonal to all the columns of $\boldsymbol{O}$, and that the elementwise product of every two columns of $\boldsymbol{O}$ is orthogonal to all columns in $\boldsymbol{O}$. In other words, if $\boldsymbol{O}$ is used as the design matrix for a second-order response surface, all estimates of linear, bilinear, and quadratic effects are uncorrelated with the estimates of linear effects. The same holds true for $\boldsymbol{O}^*$. Note that the elements of $\boldsymbol{O}$ and of $\boldsymbol{O}^*$ are no longer positive integers. However, each column is a permutation of the entries in the first column, hence both can be considered LHDs.

Ye (1998) also showed that the construction described above can be modified to yield additional OLHs. First, one can replace $\boldsymbol{e}$ by any of its permutations. Second, one can reverse any of the signs of any subset of columns of $\boldsymbol{O}$ or $\boldsymbol{O}^*$. The resulting arrays are all OLHs in the sense of having all the properties mentioned prior to Example 5.3.

5.3.4 Symmetric Latin Hypercube Designs

Unfortunately, OLHs exist only for very limited values of n_s, namely, $n_s = 2^m$ or $n_s = 2^m + 1$, $m \geq 2$. Ye et al. (2000) introduced a more general class of LHDs, called *symmetric* LHDs, to overcome this limitation. An LHD is called a symmetric LHD if it has the following property: in an $n_s \times d$ LHD with levels $1, 2, \ldots, n_s$, if $(a_1, a_2, \ldots, a_d)$ is one of the rows, then $(n_s + 1 - a_1, n_s + 1 - a_2, \ldots, n_s + 1 - a_d)$ must be another row. Ye et al. (2000) did not discuss the construction of symmetric LHDs, but when n_s is an *even* integer, one obtains a symmetric LHD as follows. The first row can be any $1 \times d$ vector $(a_{11}, a_{12}, \ldots, a_{1d})$ where the a_{1j} are elements of $\{1, 2, \ldots, n_s\}$. The second row is $(n_s + 1 - a_{11}, n_s + 1 - a_{12}, \ldots, n_s + 1 - a_{1d})$. The third row can be any $1 \times d$ vector $(a_{31}, a_{32}, \ldots, a_{3d})$ where a_{3j} can be any of the integers $1, 2, \ldots, n_s$ that is not equal to either a_{1j} or $n_s + 1 - a_{1j}$. The fourth row is $(n_s + 1 - a_{31}, n_s + 1 - a_{32}, \ldots, n_s + 1 - a_{3d})$. Continue on in this manner, adding the odd rows so that the entries in column j have not yet appeared in the previous rows of the column. The even rows have entries $n_s + 1$ minus the entry in the previous row.

When n_s is an *odd* integer, let the first row be $(\frac{n_s+1}{2}, \frac{n_s+1}{2}, \ldots, \frac{n_s+1}{2})$. The second row can be any $1 \times d$ vector $(a_{21}, a_{22}, \ldots, a_{2d})$ where the a_{2j} are elements of $\{1, 2, \ldots, n_s\}$ except $\frac{n_s+1}{2}$. The third row is $(n_s + 1 - a_{21}, n_s + 1 - a_{22}, \ldots, n_s + 1 - a_{2d})$. The fourth row can be any $1 \times d$ vector $(a_{41}, a_{42}, \ldots, a_{4d})$ where a_{4j} can be any of the integers $1, 2, \ldots, n_s$ that is not equal to $\frac{n_s+1}{2}$, a_{2j} or $n_s + 1 - a_{2j}$. Continue on in this manner, adding the even rows so that the entries in column j have not yet appeared in the previous rows of the column. The odd rows have entries $n_s + 1$ minus the entry in the previous row.

Note that the non-space-filling LHD in Fig. 5.3 is a symmetric LHD, so symmetric LHDs need not be "good" LHDs.

Example 5.6. To construct a symmetric LHD with $n_s = 10$ (an even integer) and $d = 3$, suppose the process is started with the row $(1, 6, 6)$. Following the algorithm described previously, one might obtain the following symmetric LHD,

$$\begin{pmatrix} 1 & 6 & 6 \\ 10 & 5 & 5 \\ 2 & 2 & 3 \\ 9 & 9 & 8 \\ 3 & 1 & 9 \\ 8 & 10 & 2 \\ 4 & 3 & 4 \\ 7 & 8 & 7 \\ 5 & 7 & 1 \\ 6 & 4 & 10 \end{pmatrix}.$$

To construct a symmetric LHD with $n_s = 9$ (an odd integer) and $d = 3$, suppose one begins with rows $(5, 5, 5)$ and $(1, 6, 6)$. Following the algorithm described previously, one might obtain the following symmetric LHD,

$$\begin{pmatrix} 5 & 5 & 5 \\ 1 & 6 & 6 \\ 9 & 4 & 4 \\ 2 & 2 & 3 \\ 8 & 8 & 7 \\ 3 & 1 & 9 \\ 7 & 9 & 1 \\ 4 & 3 & 8 \\ 6 & 7 & 2 \end{pmatrix}.$$

♦

Ye et al. (2000) pointed out that symmetric LHDs have certain orthogonality properties. In a polynomial response surface, least squares estimation of the linear effect of each variable is uncorrelated with all quadratic effects and bilinear interactions (but not necessarily with the linear effects of other variables). This follows from results in Ye (1998) because OLHs have the same symmetry properties as symmetric LHDs but also possess additional orthogonality that guarantees that linear effects are uncorrelated.

These orthogonality properties of OLHs and symmetric LHDs are useful if one plans to fit second-order or higher response surface models to the data using standard least squares. However, if one intends to fit a predictor, such as the EBLUP discussed in Chap. 3, in which the generalized least squares estimate of the regression parameters is used, the benefits of orthogonality are less clear.

Symmetric LHDs form a subclass of all LHDs. As noted earlier, one can apply additional criteria to select a particular design from the class of all $n_s \times d$ LHDs, from the class of all $n_s \times d$ OLHs, or from the class of all $n_s \times d$ symmetric LHDs. For the latter, Ye et al. (2000) proposed a column-wise exchange algorithm that replaces a symmetric LHD with another symmetric LHD, allowing one to search the class of $n_s \times d$ symmetric LHDs for a design that optimizes some additional property of the design.

LHDs that are optimal under an additional criterion are often symmetric LHDs. When searching for an LHD that is optimum for that criterion, restricting the search to the smaller class of symmetric LHDs often provides an easier search, and the result will often yield the global optimum over the class of all LHDs. This strategy was first proposed in Park (1994).

5.4 Designs Based on Measures of Distance

This subsection considers criteria for selecting a design that are based on a measure or metric that quantifies the spread of a set of points. While distance-based criteria can be applied without further restrictions, they can also be used to find designs within a subset of all designs. For example, one common strategy is to find an optimal distance-based design in the class of LHDs.

Let $\mathcal{X}$ denote the input space for a given simulator. For all rectangular $\mathcal{X}$, i.e,

$$\mathcal{X} = \bigtimes_{\ell=1}^{d} [a_\ell, b_\ell],$$

it will be assumed that the domain of each input has been normalized to the interval [0,1]; otherwise inputs with larger ranges can dominate the computation of a maximin design, say. Thus, the transformation

$$x_\ell = \frac{x_\ell - a_\ell}{b_\ell - a_\ell}, \quad \ell = 1, \ldots, d,$$

is used to scale and shift the input space to $[0, 1]^d$; the inverse transform is used to place the computed design on the scale of the original design problem.

Also, let $\rho_p(\cdot, \cdot)$ be a metric on $\mathcal{X}$. One important metric used in the design literature is the p^{th} order distance between $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$ which is defined by

$$\rho_p(\boldsymbol{x}_1, \boldsymbol{x}_2) = \left[\sum_{\ell=1}^{d} \left| x_{1,\ell} - x_{2,\ell} \right|^p \right]^{1/p} \tag{5.4.1}$$

for $p \geq 1$. Rectangular ("Manhattan") and Euclidean distances are the cases $p = 1$ and $p = 2$, respectively. The order p is always *assumed fixed* in the discussion below. Recall that a design, denoted $\mathcal{D}$, consisting of n_s points is a set $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{n_s}\}$ with $\boldsymbol{x}_\ell \in \mathcal{X}$, $\ell = 1, \ldots, n_s$.

The first way in which points in $\mathcal{D}$ might be regarded as spread over $\mathcal{X}$ is to require that every point in $\mathcal{X}$ be as close as possible to a point in $\mathcal{D}$. This notion is made precise by defining the distance between an arbitrary input site $\boldsymbol{x} \in \mathcal{X}$ and $\mathcal{D}$ to be

$$\rho_p(\boldsymbol{x}, \mathcal{D}) = \min_{\boldsymbol{x}_i \in \mathcal{D}} \rho_p(\boldsymbol{x}, \boldsymbol{x}_i)$$

which is the closest any point in $\mathcal{D}$ is to $\boldsymbol{x}$. An n_s-point design $\mathcal{D}_{mM}$ is defined to be a *minimax distance design* if the maximum distance between arbitrary points $\boldsymbol{x} \in \mathcal{X}$ and the candidate design $\mathcal{D}_{mM}$ is a minimum over all designs $\mathcal{D}$ whose input vectors $\boldsymbol{x}_\ell \in \mathcal{X}$, $\ell = 1, \ldots, n_s$; namely,

$$\max_{\boldsymbol{x} \in \mathcal{X}} \rho_p(\boldsymbol{x}, \mathcal{D}_{mM}) = \min_{\mathcal{D}} \max_{\boldsymbol{x} \in \mathcal{X}} \rho_p(\boldsymbol{x}, \mathcal{D}).$$

If the goal of a computer experiment is good prediction over all of $\mathcal{X}$, and if prediction variance at a point $\boldsymbol{x}_0$ increases as the distance between $\boldsymbol{x}_0$ and $\mathcal{D}$ increases, intuitively a design $\mathcal{D}$ for which no point is far from any $\boldsymbol{x}_0 \in \mathcal{X}$ should perform well. In other words, a minimax design would seem to be a sensible choice if the goal is good prediction (minimizing the maximum prediction variance) over $\mathcal{X}$. The difficulty is that finding minimax designs involves computing the maximum distance between a candidate design $\mathcal{D}$ and all points in $\mathcal{X}$. This is computationally challenging. One might try to find an approximately minimax design by restricting the computation to a finite grid of points in $\mathcal{X}$. See Tan (2013) for a discussion of this approach.

A second method to measure the spread of the n_s points in a design is by the distance of the closest two points in the design, i.e., by

$$\min_{\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{D}} \rho_p(\boldsymbol{x}_1, \boldsymbol{x}_2). \tag{5.4.2}$$

A design that maximizes (5.4.2) is said to be a *maximin distance design* and is denoted by $\mathcal{D}_{Mm}$; thus

$$\min_{\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{D}_{Mm}} \rho_p(\boldsymbol{x}_1, \boldsymbol{x}_2) = \max_{\mathcal{D} \subset \mathcal{X}} \min_{\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{D}} \rho_p(\boldsymbol{x}_1, \boldsymbol{x}_2). \tag{5.4.3}$$

In an intuitive sense, therefore, $\mathcal{D}_{Mm}$ designs guarantee that no two points in the design are too close, and hence the design points are spread over $\mathcal{X}$.

One criticism of the maximin principle is that it judges the goodness of a design by the minimum distance between all $\binom{n_s}{2}$ pairs of input vectors rather than using all possible differences. Figure 5.7 illustrates such a pair of designs both of which have as their three smallest minimum interpoint distances 0.300, 0.361, and 0.412. The only difference in the two designs is that the point $(0.2, 0.2)$ in the left panel design has been moved to $(0.025, 0.025)$ in the right panel, but because of this change, the design in the right panel is, intuitively, more space-filling than the design in the left panel. More careful inspection of these designs shows that the fourth smallest interpoint distance is greater for the right panel design than the left panel design. By using a more comprehensive definition of minimaxity in which the number of pairs of the inputs with smallest, second smallest, etc. distances are accounted for, Morris and Mitchell (1995) were able to rank cases of equal minimum inter-point distance and eliminate such anomalies. In sum, despite this initial criticism, Mm designs are often visually attractive and can be justified theoretically under certain circumstances (Johnson et al. (1990)).

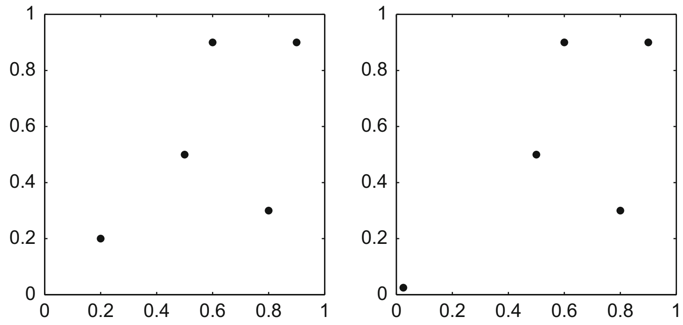


Fig. 5.7 Two designs on $[0, 1]^2$ with the same minimum interpoint distance of 0.30

Mm designs on $[0, 1]^d$ can be computed by solving the mathematical programming problem

$$\begin{gathered} \max z \\ \text{subject to} \\ z \le \rho_p(\boldsymbol{x}_i, \boldsymbol{x}_j)\,, \quad 1 \le i < j \le n_s \\ \mathbf{0}_d \le \boldsymbol{x}_\ell \le \mathbf{1}_d, \quad 1 \le \ell \le n_s \end{gathered} \tag{5.4.4}$$

in which an additional decision variable z has been added to the unknown $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n_s}$; z is a lower bound for all distances in (5.4.4). While this problem can be solved by standard nonlinear programming algorithms for "small" n_s, the computational difficulty with this approach is that the number of constraints on z grows on the order of n_s^2 (see Stinstra et al. (2003)).

Example 5.7. Figure 5.8 displays four-point maximin and minimax designs with Euclidean distance ($p = 2$ in (5.4.1)). The differences in the two designs reflect the effect of the different design criteria. Maximin designs tend to push points out toward the boundary of $\mathcal{X}$. This is not surprising because there is more "space" to spread out points at the boundary. ♦

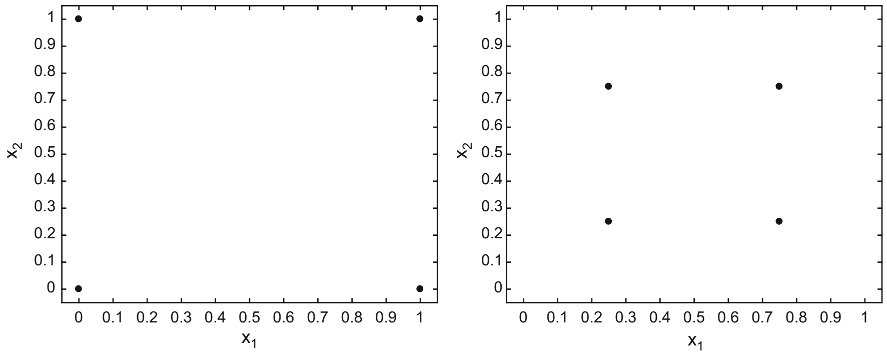


Fig. 5.8 Left panel: a four-point maximin design on $[0, 1]^2$ with respect to Euclidean distance; right panel: a four-point minimax design on $[0, 1]^2$

A third approach to spreading points in the design space is to consider the *distribution of distances* between *all pairs of input vectors* and not merely the distance between the *closest* pair of input vectors. One example of a criterion that uses this approach chooses the design to minimize the average of the reciprocals of the distances between all pairs of design points, the so-called average reciprocal distance (ARD) of the design. The ARD of design $\mathcal{D}$ is defined to be

$$m_{(p,\lambda)}(\mathcal{D}) = \left(\frac{1}{\binom{n_s}{2}} \sum_{\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}} \left[\frac{1}{\rho_p(\boldsymbol{x}_i, \boldsymbol{x}_j)} \right]^{\lambda} \right)^{1/\lambda}, \quad \lambda \geq 1\,. \tag{5.4.5}$$

The combinatorial coefficient $\binom{n_s}{2}$ is the number of different pairs of rows in the design. For example, when $\lambda = 1$, the criterion function $m_{(p,1)}(\mathcal{D})$ is inversely proportional to the harmonic mean of the distances between design points.

For fixed (p, λ), an $n_s \times d$ design $\mathcal{D}_{av}$ is a *minimum ARD* (mARD) design if

$$m_{(p,\lambda)}(\mathcal{D}_{av}) = \min_{\mathcal{D} \subset \mathcal{X}} m_{(p,\lambda)}(\mathcal{D})\,. \tag{5.4.6}$$

The optimality condition (5.4.6) favors designs that possess nonredundancy in the location of input sites; specifically the criterion does not allow design points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ that are (simultaneously) the same in *all* coordinates, i.e., with $\boldsymbol{x}_i = \boldsymbol{x}_j$. When $\lambda =$ 1, the optimality condition (5.4.6) selects designs which maximize this harmonic mean, of course, preventing any "clumping" of design points. The nonredundancy requirement can be seen even more clearly for large values of λ. Taking $\lambda \to \infty$, the criterion function (5.4.5) limit is

$$m_{(p,\infty)}(\mathcal{D}) = \max_{\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}} \frac{1}{\rho_p(\boldsymbol{x}_i, \boldsymbol{x}_j)}\,. \tag{5.4.7}$$

Minimizing the right hand side of (5.4.7) is equivalent to maximizing (5.4.2). Thus, an n_s-point design $\mathcal{D}_{Mm}$ satisfying condition (5.4.6) for the limiting distances as $\lambda \to \infty$, namely,

$$m_{(p,\infty)}(\mathcal{D}_{Mm}) = \min_{\mathcal{D} \subset \mathcal{X}} m_{(p,\infty)}(\mathcal{D})\,,$$

is a maximin distance design as defined previously:

$$\max_{\mathcal{D} \subset \mathcal{X}} \min_{\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}} \rho_p(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{1}{m_{(p,\infty)}(\mathcal{D}_{Mm})}\,.$$

Example 5.8. Figure 5.9 displays minimum ARD designs on $[0,1]^2$ for $(n_s, d) =$ $(6, 2)$ with Euclidean distance for $\lambda = 1$ and for $\lambda = \infty$; by (5.4.7) the $\lambda = \infty$ design is a Mm design for this Euclidean distance case. Both designs concentrate points on or near the boundary of $[0,1]^2$ so that the projections of the design points onto either axis produces multiple observations in 1-D. If the output depends primarily on one of the inputs, say x_1, this means that such a design will not fully explore x_1 space. This defect can be remedied by restricting the class of available designs to include only, say, LHDs. Figure 5.2 is an example of a mARD within the class of LHDs for $p = 1 = \lambda$. ♦

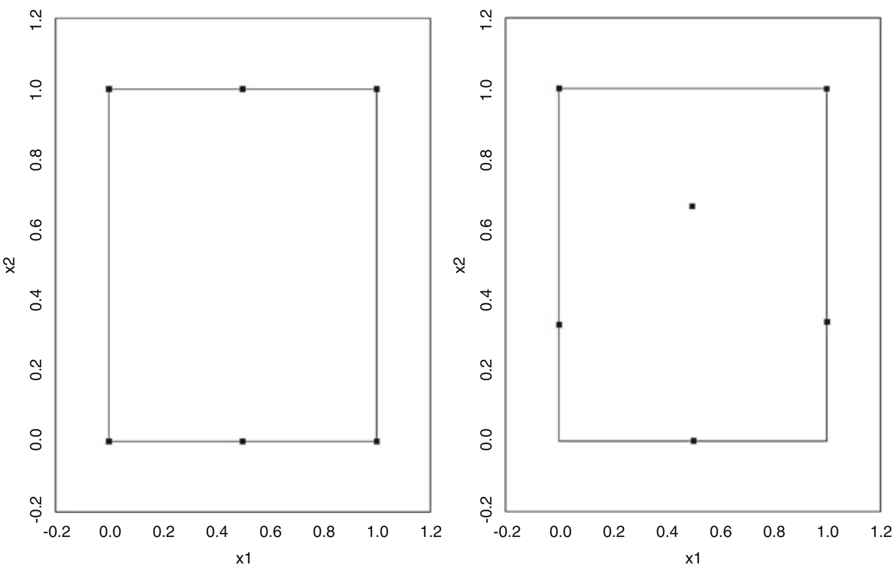


Fig. 5.9 Minimum ARD designs on $[0, 1]^2$ with respect to Euclidean distance for $p = 2$ and left panel: $\lambda = 1.0$; right panel: $\lambda = \infty$

As noted above, neither the Mm nor the mARD optimal designs need have projections onto spaces of dimension $< d$ which are nonredundant. To reiterate, consider a computer experiment involving $d = 5$ inputs, only three of which (say) are active. In this event, a desirable property of an optimal $n_s \times 5$ design is nonredundancy of input sites projected onto the three-dimensional subspace of the active inputs. Such designs can be generated by computing the criterion values (5.4.5) for each relevant projection of the full design $\mathcal{D}$ and averaging these to form a new criterion function which is then minimized by choice of design $\mathcal{D}$. The approach is implemented by the *Algorithms for the Construction of Experimental Designs* (ACED) software of Welch (1985), among other packages. The Welch (1985) software was used to compute the optimal designs of this section.

Formally, one version of the projection approach sketched in the previous paragraph can be described as follows. Let $J \subseteq \{1, 2, \ldots, d\}$ denote the set of subspace dimensions for which nonredundancy of input sites is desired. For example, if $2 \in J$, then it is desired that projecting the design onto every 2-D subspace of $[0, 1]^d$ results in a space-filling set of points. For each $j \in J$, let $\{S_{kj}\}$ denote the k^{th} design in an enumeration of all j-dimensional projections of $\mathcal{D}$ for $k = 1, \ldots, \binom{d}{j}$. Because the maximum distance between any two points in $[0, 1]^d$ is $j^{1/p}$, it is essential that the $\rho_p(\cdot, \cdot)$ distance between points in a j-dimensional projection be normalized by this maximum distance in order for distances across different dimensional projections to be comparable. For $k = 1, \ldots, \binom{d}{2}$ and $j \in J$ define the minimum distance for the projected design $\mathcal{D}_{kj}$ to be

$$\min_{\boldsymbol{x}_h^\star, \boldsymbol{x}_\ell^\star \in \mathcal{D}_{kj}} \frac{\rho_p(\boldsymbol{x}_h^\star, \boldsymbol{x}_\ell^\star)}{j^{1/p}}$$

where $\boldsymbol{x}_h^\star$ and $\boldsymbol{x}_\ell^\star$ denote the projections of $\boldsymbol{x}_h$ and $\boldsymbol{x}_\ell$ onto the subspace determined by j and k. Given p define the J-minimum of the design $\mathcal{D}$ to be

$$\rho_J(\mathcal{D}) = \min_{j \in J} \min_{k \in \{1,\ldots,\binom{d}{j}\}} \min_{\boldsymbol{x}_h^\star, \boldsymbol{x}_\ell^\star \in \mathcal{D}_{kj}} \frac{\rho_p(\boldsymbol{x}_h^\star, \boldsymbol{x}_\ell^\star)}{j^{1/p}} \tag{5.4.8}$$

and given p and λ define the J-average reciprocal distance to be

$$av_J(\mathcal{D}) = \left(\frac{1}{\binom{n_s}{2} \times \sum_{j \in J} \binom{d}{j}} \sum_{j \in J} \sum_{k=1}^{\binom{d}{j}} \sum_{\boldsymbol{x}_h^*, \boldsymbol{x}_\ell^* \in \mathcal{D}_{kj}} \left[\frac{j^{1/p}}{\rho_p(\boldsymbol{x}_h^\star, \boldsymbol{x}_\ell^\star)} \right]^\lambda \right)^{1/\lambda} . \tag{5.4.9}$$

An n_s-point design $\mathcal{D}_{MmP}$ is said to be a *J-maximin design* with respect the projection criterion (5.4.8) provided

$$\rho_J(\mathcal{D}_{MmP}) = \max_{\mathcal{D}} \rho_J(\mathcal{D}),$$

while $\mathcal{D}_{avp}$ is said to be a *J-minimum ARD design* with respect to the projection criterion (5.4.9) if

$$\text{av}_J(\mathcal{D}_{avp}) = \min_{\mathcal{D} \subset \mathcal{X}} \text{av}_J(\mathcal{D}) . \tag{5.4.10}$$

As for maximin and minimum ARD designs, the optimal J-maximin and J-minimum ARD designs (5.4.10) will also be more nearly space-filling if the class of designs searched is restricted to LHDs.

Example 5.9. As an example, let $(n_s, d) = (10, 3)$ and fix $p = \lambda = 1$. A J-minimum ARD design in the class of LHDs was generated for $J = \{2, 3\}$. Figure 5.10 presents a 3-D scatterplot of the design and 2-D scatterplot of the projection of the design onto the (x_2, x_3) subspace. Note that, because LHDs are nonredundant in each one-dimensional subspace by definition, $1 \notin J$. ♦

The final example of a distance-based criterion that will be described in this book is the *maximum projection* (MaxPro) *design* introduced by Joseph et al. (2015). As do J-maximin and J-minimum ARD designs, MaxPro designs take into account projections. As for J-maximin and J-minimum ARD designs, the MaxPro design criterion can be used to construct a design from scratch or to augment a given design.

MaxPro designs are defined in terms of a criterion which is a *weighted* version of the pth order metric (5.4.1). The weighted pth order metric is defined by

$$\rho_p(\boldsymbol{x}_1, \boldsymbol{x}_2; \boldsymbol{w}) = \left[\sum_{j=1}^{d} w_j \left| x_{1,j} - x_{2,j} \right|^p \right]^{1/p}$$

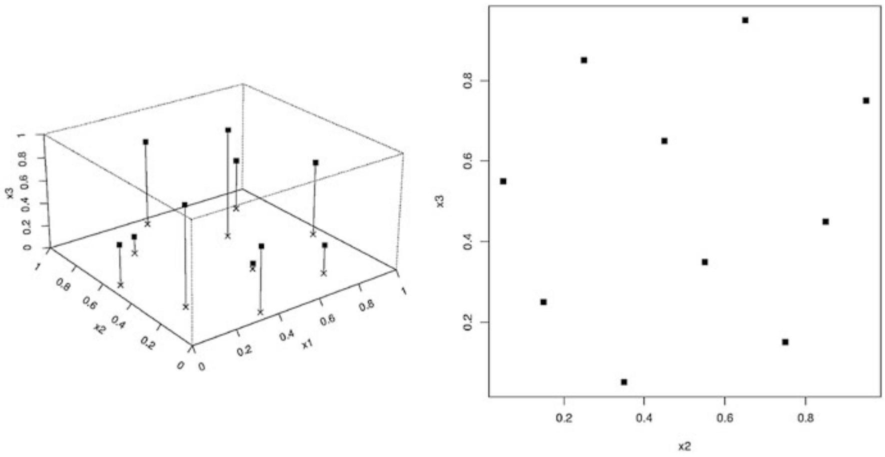


Fig. 5.10 Left panel: a 3-D plot of a $n_s = 10$ point J-minimum ARD within the class of LHDs when $p = \lambda = 1$ and $J = \{2, 3\}$. Right panel: projection of left panel design onto x_2-x_3 plane

for $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ in $\mathcal{X}$. For example, if $w_1 = 1$ and $w_j = 0$, $j \geq 2$, then $\rho_p(\boldsymbol{x}_1, \boldsymbol{x}_2; \boldsymbol{w})$ is the p^{th} order metric for the projection onto the x_1 subspace. Similarly, if $w_1 = 1/2 = w_2$ while $w_j = 0$, $j \geq 3$, then $\rho_p(\boldsymbol{x}_1, \boldsymbol{x}_2; \boldsymbol{w})$ is $(0.5\ \times)$ the p^{th} order metric for the projection onto the (x_1, x_2) subspace.

Now suppose that $\pi(\boldsymbol{w})$ is a given distribution that describes the importance of the weights $\boldsymbol{w} = (w_1, w_2, \ldots, w_d)$, where $\boldsymbol{w}$ is constrained to the $(d-1)$-dimensional simplex

$$\mathcal{S}_{d-1} = \left\{ \boldsymbol{w} \,:\, w_1, \ldots, w_{d-1} \geq 0, \sum_{j=1}^{d-1} w_j \leq 1 \right\}$$

and $w_d = 1 - \sum_{j=1}^{d-1} w_j$. The MaxPro criterion chooses $\mathcal{D}$ to minimize

$$\int_{\mathcal{S}_{d-1}} \sum_{\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}: i \neq j} \frac{1}{\rho_p^k(\boldsymbol{x}_i, \boldsymbol{x}_j; \boldsymbol{w})} \pi(\boldsymbol{w})\, d\boldsymbol{w}\,. \tag{5.4.11}$$

Unfortunately the criterion (5.4.11) can be computationally demanding; hence Joseph et al. (2015) recommend using a uniform (on the simplex) weight function $\pi(\boldsymbol{w})$, Euclidean distance ($p = 2$), and $k = p \times d$ for which they show (5.4.11) is, aside from constants,

$$\sum_{\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}: i \neq j} \frac{1}{\prod_{\ell=1}^{d} (x_{i,\ell} - x_{j,\ell})^2}\,,$$

which can be computed rapidly.

Example 5.10 (Comparison of a Maximum Projection Design and a J-Minimum ARD). Consider an experiment with $d = 4$ inputs that is to be run using $n_s = 10$ points. Table 5.1 lists the MaxPro 10×4 design constructed using the R software

MaxPro and the J-minimum ARD with $J = \{1, 2\}$, $p = 2$, and $\lambda = 1$ that was constructed using the R software concad. The MaxPro design is meant to cover all possible projections and hence can be appropriate for simulator outputs having arbitrary numbers of interacting inputs, while the J-minimum ARD design is meant to provide well-spread projections for simulator output that contains up to 2-D interactions. Figures 5.11 and 5.12 are scatterplot matrices of all 2-D projections and histograms of the 1-D projections for the two designs. Both designs appear well-spread over all six 2-D projections although the J-minimum ARD emphasizes the edges of the design variables a bit more than the MaxPro design.

MaxPro design				J-minimum ARD			
x_1	x_2	x_3	x_4	x_1	x_2	x_3	x_4
0.05	0.75	0.45	0.15	0.00	1.00	0.67	0.63
0.15	0.15	0.55	0.75	0.94	0.07	1.00	0.00
0.25	0.55	0.05	0.45	0.62	0.37	0.00	1.00
0.35	0.65	0.85	0.95	0.29	0.62	0.16	0.22
0.45	0.35	0.65	0.05	0.81	0.79	0.44	0.84
0.55	0.95	0.25	0.65	0.17	0.24	0.86	0.42
0.65	0.25	0.95	0.55	0.44	0.01	0.31	0.73
0.75	0.05	0.15	0.25	0.72	0.92	0.93	0.11
0.85	0.85	0.75	0.35	0.99	0.56	0.56	0.51
0.95	0.45	0.35	0.85	0.08	0.70	0.06	0.05

Table 5.1 A 10×4 MaxPro design (left four columns) and 10×4 J-minimum ARD design with $J = \{1, 2\}$ and $(p, \lambda) = (2, 1)$ (right four columns)

In addition to a comparison by their visual appearance, the designs can be compared numerically in a large number of ways, most of which the designs were *not* constructed to optimize. One exception to the previous sentence is a comparison using the ARD in (5.4.10) when $(p, \lambda) = (2, 1)$ and $J = \{1, 2\}$; for this criterion the J-minimum ARD design achieves the ARD of 3.260 (and 4.132 and 3.006 for $J = \{1\}$, $J = \{1, 2, 3\}$, respectively), while the MaxPro design has ARD of 3.501 (and 4.287 and 3.245 for $J = \{1\}$, $J = \{1, 2, 3\}$). This is to be expected because (5.4.10) is the criterion to be minimized by the J-minimum ARD design. The comparison shows that the MaxPro design is 93% (= 3.260/3.501) as efficient as the design specifically created for the ARD criterion.

Other distance metrics provide a different picture. Comparing the minimum pairwise row distances for any projection requires calculating a minimum distance over $\binom{10}{2} = 45$ pairs of rows. Intuitively minimal point distances which are *large* suggest a better design. The minimum 4-D pairwise row distance for the MaxPro design is 0.574 which is larger (better) than the 0.299 minimum 4-D distance for the J-minimum ARD design. There are four 3-D projections for each design (onto the dimensions $(x_1, x_2, x_3), \ldots, (x_2, x_3, x_4)$) which are listed in Table 5.2. The MaxPro design maximizes all four projections. Similarly there are six possible (and minimal) 2-D projections for each design which are listed in Table 5.3. Half are larger for each design.

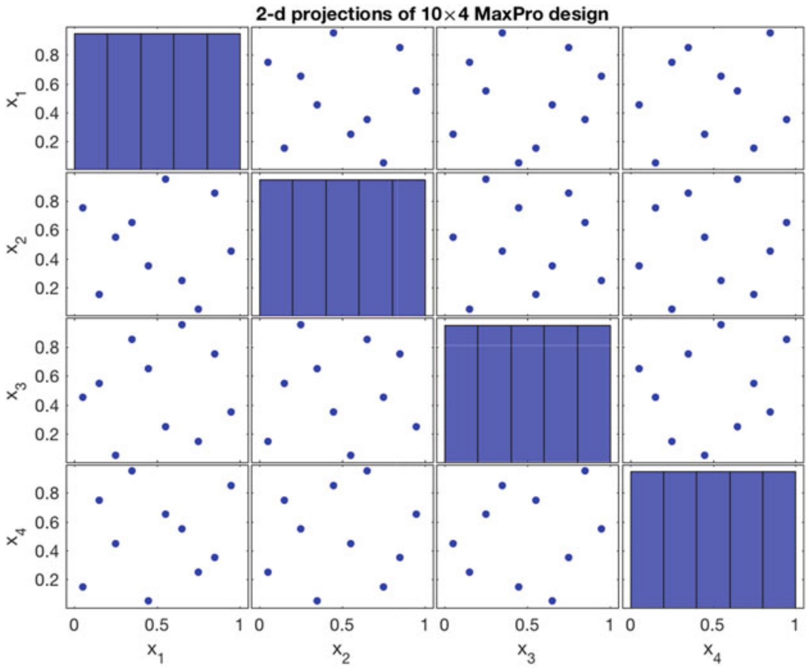


Fig. 5.11 Scatterplot matrix of a 10×4 MaxPro design

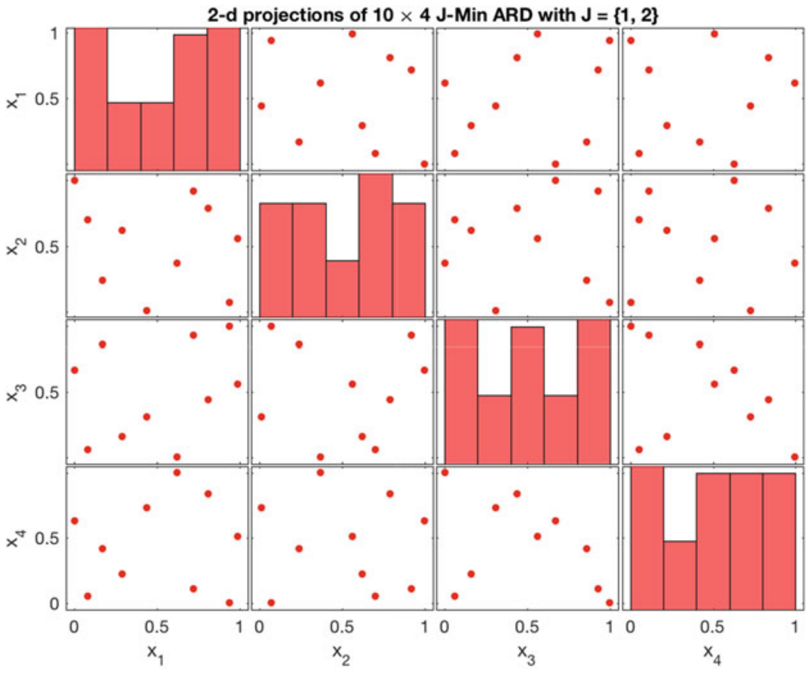


Fig. 5.12 Scatterplot matrix of a J-minimum ARD with $J = \{1, 2\}$

Of course, the bottom line of design is to facilitate the accurate prediction of simulator output functions having certain characteristics. In some application areas, this comparison may be best assessed by a simulation study that empirically assesses prediction accuracy for functions having the desired attributes. ♦

Col	Col	Col	MaxPro design	J-minimum ARD design
1	2	3	**0.374**	0.246
1	2	4	**0.374**	0.282
1	3	4	**0.346**	0.256
2	3	4	**0.374**	0.213

Table 5.2 Minimum Euclidean distances for all 3-D projections of the 10×4 MaxPro and the J-minimum ARD designs with $J = \{1, 2\}$; the maximum value for each projection is in **bold face**

Col	Col	MaxPro design	J-minimum ARD design
1	2	0.141	**0.158**
1	3	0.141	**0.212**
1	4	0.141	**0.233**
2	3	**0.224**	0.128
2	4	**0.224**	0.129
3	4	**0.224**	0.130

Table 5.3 Minimum pairwise Euclidean distances for all 2-D projections of the 10×4 MaxPro and the J-minimum ARD designs with $J = \{1, 2\}$; the maximum value for each projection is in **bold face**

One final note when using distance-based criteria concerns the choice of metric. Euclidean distance is a common choice. For the GP model, Euclidean distance is reasonable if the model is isotropic or if there is no prior information about the relative sizes of the correlation parameters. However, if there is prior information about the correlation parameters, Mahalanobis distance or some sort of weighted distance, with weights determined by the correlation parameters, may be more appropriate. See Williams et al. (2011) where the use of Mahalanobis distance is considered in the context of a sequential design strategy.

5.5 Distance-Based Designs for Non-rectangular Regions

Sections 5.2–5.4 described criteria for constructing space-filling designs when the input region is hyper-rectangular. However non-rectangular input regions occur naturally in many applications when the range of one or more inputs depends on the values of other inputs. As an example, Hayeck (2009) studied the effects of four input variables on the functioning of a total elbow prosthesis using a simulator model.

One input was a biomechanical engineering design variable, and three were environmental variables. The biomechanical variable was the tip displacement (x_1, in *mm*), and the environmental variables were the rotation, at the tip, of the implant axis about the lateral axis (x_2 in degrees), the rotation of the implant axis, at the tip, about the anterior axis (x_3 in degrees), and the rotation about the implant axis (x_4 in degrees). The following constraints were imposed on the four input variables based on anatomical considerations

$$\begin{array}{rcl} 0 \leq & x_1 & \leq 10 \\ -10 \leq & 5x_2 + 2x_3 & \leq 10 \\ -10 \leq & -5x_2 + 2x_3 & \leq 10 \\ -15 \leq & x_4 & \leq 15\,. \end{array} \tag{5.5.1}$$

These constraints state, among other things, that the maximum tip displacement is 10 mm and the rotation about the implant axis is $\pm 15°$. The outputs of the computational simulation were various stresses and strains in the elbow.

This section focusses on finding (approximate) *maximin distance* (Mm) and the *minimum ARD* (mARD) designs for non-rectangular input regions (see (5.4.3) and (5.4.6), respectively). For definiteness, the majority of this section restricts attention to input regions that are bounded polytopes, i.e., to designs

$$\mathcal{D} = \begin{pmatrix} \boldsymbol{x}_1^\top \\ \vdots \\ \boldsymbol{x}_{n_s}^\top \end{pmatrix}$$

where

$$\boldsymbol{A}\boldsymbol{x}_i \leq \boldsymbol{b}\,, \tag{5.5.2}$$

for $i = 1, \ldots, n_s$ and $\boldsymbol{A}$ and $\boldsymbol{b}$ are known. Bounds on individual variables, e.g., that $0 \leq x_1 \leq 1$, are assumed to be included in the polytope constraints.

For example, the Hayeck (2009) input region (5.5.1) is the polytope defined by

$$\boldsymbol{A} = \begin{pmatrix} +1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & +5 & +2 & 0 \\ 0 & -5 & -2 & 0 \\ 0 & -5 & +2 & 0 \\ 0 & +5 & -2 & 0 \\ 0 & 0 & 0 & +1 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

and

$$\boldsymbol{b} = (10, 0, 10, 10, 10, 10, 15, 15)^\top\,.$$

Before describing several of the approaches that have been proposed in the literature for finding Mm and mARD designs, we reiterate the recommendation made in Sect. 5.4 that, because inputs with different scales can cause the computation of a Mm design to be dominated by those inputs having larger ranges, all inputs in a bounded non-rectangular problem be scaled and shifted to the interval [0,1]. For example, for the bounded polytope $\boldsymbol{Ax} \le \boldsymbol{b}$, the j^{th} input has maximum and minimum which are the solutions to

$$\max x_j \quad \text{subject to} \quad \boldsymbol{Ax} \le \boldsymbol{b} \quad \text{and} \quad \min x_j \quad \text{subject to} \quad \boldsymbol{Ax} \le \boldsymbol{b}\,.$$

Second, maximin designs need not have "space-filling" projections onto subsets of the input variables although selecting designs using the J-maximin or J-minimum ARD criterion or from the class of LHDs can (partially) eliminate this problem.

In principle, maximin designs for the case of inputs that satisfy (5.5.2) can be solved by modifying the mathematical programming problem (5.4.4) to

$$\begin{gathered}\max z\\ \text{subject to}\\ z \le \rho_2(\boldsymbol{x}_i, \boldsymbol{x}_j)\,, \quad 1 \le i < j \le n_s\\ \boldsymbol{A}\boldsymbol{x}_\ell \le \boldsymbol{b}\,, \quad 1 \le \ell \le n_s\,.\end{gathered}$$

Other constraints on the $\boldsymbol{x}_\ell$ can be handled similarly. However the simultaneous solution for $\mathcal{D}$ becomes prohibitive as $n_s \times d$ grows.

In the spirit of Morris and Mitchell (1995), Trosset (1999) replaced the pairwise row minimum in (5.4.3) by

$$\max_{i<j} \phi(\rho_p(\boldsymbol{x}_i, \boldsymbol{x}_j)) \tag{5.5.3}$$

where $\phi(w)$ is a strictly decreasing function such as $\phi(w) = 1/w$. For large λ, a design that maximizes

$$\left\{ \sum_{i<j} \phi\left(\rho_2(\boldsymbol{x}_i, \boldsymbol{x}_j)\right)^{\lambda} \right\}^{1/\lambda} \tag{5.5.4}$$

subject to $\boldsymbol{Ax} \le \boldsymbol{b}$ is an approximate maximin design subject to the polytope constraint because (5.5.4) converges to (5.5.3) as $\lambda \to \infty$.

Stinstra et al. (2003) introduced a mathematical program for finding maximin designs and an algorithm for approximately solving the mathematical programming problem; the formulation can allow non-rectangular input regions. Their algorithm solves a sequence of n_s subproblems to update a current feasible set of points $\boldsymbol{x}^c = \left(\boldsymbol{x}_1^c, \ldots, \boldsymbol{x}_{n_s}^c\right)$ in the order $\boldsymbol{x}_1^c, \ldots, \boldsymbol{x}_{n_s}^c$. Fix i with $1 < i < n_s$ and assume that $\boldsymbol{x}_\ell^c$ has been previously updated to $\boldsymbol{x}_\ell^{c+1}$, for $1 \le \ell < i$. Then $\boldsymbol{x}_i^c$ is determined by solving

$$\begin{gathered}\max w\\ \text{subject to}\\ w \le \rho_2(\boldsymbol{x}, \boldsymbol{x}_\ell^{c+1})\,, \quad \ell < i\\ w \le \rho_2(\boldsymbol{x}, \boldsymbol{x}_\ell^{c})\,, \quad \ell > i\\ \boldsymbol{A}\boldsymbol{x}_i \le \boldsymbol{b}\end{gathered}$$

for $(w^\star, \boldsymbol{x}^\star)$ and setting $\boldsymbol{x}_i^{c+1} = \boldsymbol{x}^\star$. The algorithm is modified in an obvious way for $i \in \{1, n_s\}$. This cycle of n_s steps is repeated until a given minimum improvement in (5.4.1) occurs or a computational budget is exhausted.

In contrast to the row-by-row construction of a Mm design used by Stinstra et al. (2003), Draguljić et al. (2012) employed a column-by-column construction of $\mathcal{D}$ to form space-filling designs for input regions that are bounded polytopes. Their algorithm allows user specified criteria; the R program concad of Draguljić (see Sect. 5.7) allows the Mm, mARD, or a criteria that combines the Mm and mARD metrics. Their designs also satisfy the requirement that the design is "non-collapsing" in each input. A design is non-collapsing when the projection of the design onto each input has no exact duplicates.

Lastly, the method of Tan (2013) for finding minimax designs over finite design spaces, mentioned in Sect. 5.4, includes situations in which the finite design space appears non-rectangular.

5.6 Other Space-Filling Designs

5.6.1 Designs Obtained from Quasi-Random Sequences

Quasi-random sequences are intended to produce finite sequences of points that fill the d-dimensional unit hypercube and have the property that a design with sample size n_s is obtained from the design of sample size $n_s - 1$ by adding a point to the design. Although introduced for numerically evaluating multidimensional integrals, they also allow one to generate space-filling designs.

Several such sequences have been proposed, including Halton sequences (Halton (1960)), Sobol´ sequences (Sobol´ (1967, 1976)), and Niederreiter sequences (Niederreiter (1988)). Appendix E presents some details for constructing the simplest of these sequences, the Halton sequence, as well as Sobol´ sequences.

Example 5.11. In Sect. 3.3, Sobol´ sequences were used to select the values of the environmental variables and compared to other methods. The left-hand panel of Fig. 5.13 displays *1* of the 6, 2-D projections of the 40-point Sobol´ sequence in the standardized $d = 4$ variables that were used as inputs to generate the fire containment data of Example 1.1; see Fig. 3.2 for the complete set of 6 2-D projections. The corresponding 2-D projection for the same input pair of the 40-point *maximin LHD* is shown in the right-hand panel of the same figure. It is clear from Fig. 5.13 that the maximin LHD has projections which are spread more evenly than the design based on the Sobol´ sequence. Thus, if it is important that the design be evenly spread out, the LHD appears to be preferable. On the other hand, the design based on the Sobol´ sequence appears to exhibit a greater variety of interpoint distances (distances between pairs of points in the design) than the LHD. If a greater variety of interpoint distances provides more information about the correlation parameters (and hence allows one to better estimate these parameters), then designs based on a Sobol´ sequence (or other types of sequences that have been used in numerical integration) may be preferable to the LHD. ♦

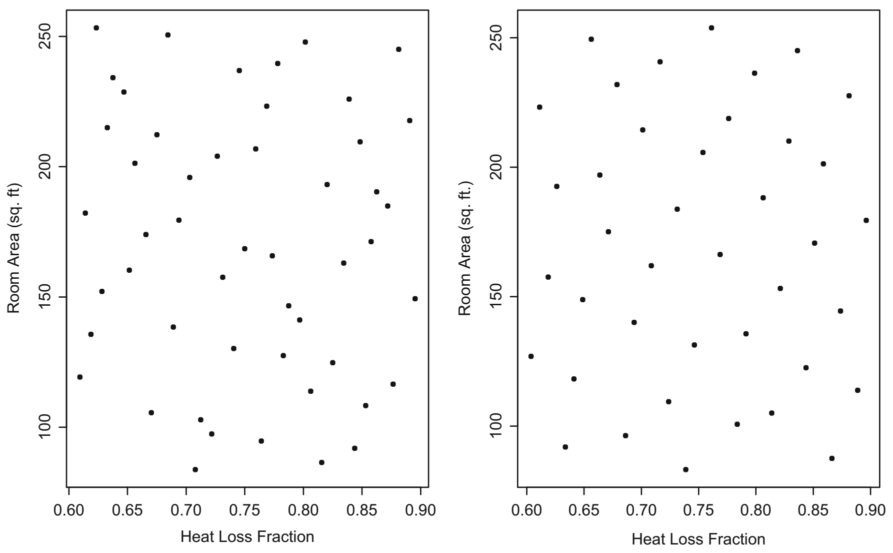


Fig. 5.13 Left panel: 2-D projection of the 40-point Sobol´ sequence on $[0, 1]^4$ (when standardized, these two inputs are to be used as the inputs for room area × heat loss fraction); right panel: 2-D projection of the 40-point maximin LHD on $[0, 1]^4$ (the standardized values of these inputs are used as the inputs for room area × heat loss fraction)

Suppose one uses a space-filling design consisting of n_s points in the unit cube. After fitting a predictor to the data generated by the simulator, suppose one decides the fit is inadequate and m_s additional runs of the computer simulator are necessary. Is it possible to select the m_s runs in such a way that the resulting set of $n_s + m_s$ runs is space-filling?

In Sect. 5.2 we saw that for LHDs this is only possible in special cases. However, because of the method of construction, this is possible for designs generated by Halton, Sobol´, and Niederreiter sequences. Thus, if the initial n_s design consists of the first n_s points in one of these sequences, simply add the next m_s points in the sequence to generate the larger design. To the extent that Halton, Sobol´, and Niederreiter sequences are space-filling, both the initial and final designs will also be space-filling (although the degree to which the designs look space-filling will depend on the particular values of n_s and m_s).

This ability to add points so that both the initial and final design are reasonably space-filling makes quasi-random sequences such as Halton, Sobol´, and Niederreiter sequences appear attractive in the context of sequential experimentation. However, quasi-random sequences are usually space-filling only in the sense described in Sect. 5.1, namely, as the number of points in the sequence increases, the sequence becomes increasingly dense in the design space (here assumed to be the

d-dimensional unit cube). As Fig. 5.13 suggests, quasi-random sequences need not look particularly space-filling for small to moderate sample sizes. Furthermore, if the number of runs is not a power of 2 (assuming the common case of base 2 for the construction of the sequence described in Appendix E), then subsequent points do not necessarily fill in the most empty part of input space. Finally, such sequences can have bad projection properties. Liefvendahl and Stocki (2006) show in some analytic test problems that the statistical accuracy of predictors based on designs generated by the minimum ARD criterion is superior to that based on designs produced by a Sobol´ sequence. For these reasons, designs based on quasi-random sequences are much less popular in practice than other space-filling designs.

5.6.2 Uniform Designs

In Sect. 5.2 we considered criteria for selecting a space-filling design based on sampling methods and, in Sects. 5.4 and 5.5, criteria based on distances between points. In this section, we consider a third intuitive design principle based on comparing the distribution of the points in a design to the uniform distribution.

As in Sect. 5.2.2, suppose that the vector of inputs is d-dimensional and denoted by $\boldsymbol{x} = (x_1, \ldots, x_d)$. Also again assume that $\boldsymbol{x}$ must fall in the d-dimensional hypercube $\mathcal{X} = [0, 1]^d$, possibly after recentering and rescaling of the inputs. Let $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{n_s}\}$ denote the set of n_s points at which we will observe the response $y(\boldsymbol{x})$. If we wish to emphasize that $\boldsymbol{x}$ is a random variable, we will use the notation $\boldsymbol{X}$. This would be the case, for example, if we are interested in $\mathrm{E}[y(\boldsymbol{X})]$. Below we take $\boldsymbol{X} \sim F(\cdot)$ where

$$F(\boldsymbol{x}) = \prod_{i=1}^{d} x_i \qquad (5.6.1)$$

is the uniform distribution on $[0, 1]^d$ (other choices of distribution function are possible).

Fang et al. (2000, 2006) discuss the notion of the *discrepancy* of a design $\mathcal{D}$, which measures the extent to which $\mathcal{D}$ differs from a completely uniform distribution of points. To be specific, let F_{n_s} be the empirical distribution function of the points in $\mathcal{D}$, namely,

$$F_{n_s}(\boldsymbol{x}) = \frac{1}{n_s} \sum_{i=1}^{n_s} I\{\boldsymbol{X}_i \le \boldsymbol{x}\}, \qquad (5.6.2)$$

where $I\{E\}$ is the indicator function of the event E and the inequality is with respect to the componentwise ordering of vectors in $\mathbb{R}^d$. The L_∞ discrepancy, sometimes called *star discrepancy* or simply discrepancy, is denoted $D_\infty(\mathcal{D})$ and is defined as

$$D_\infty(\mathcal{D}) = \sup_{\boldsymbol{x} \in \mathcal{X}} |\, F_{n_s}(\boldsymbol{x}) - F(\boldsymbol{x}) \,| \,. \qquad (5.6.3)$$

This is perhaps the most popular measure of discrepancy and is the Kolmogorov–Smirnov statistic for testing fit to the uniform distribution.

Example 5.12. Suppose $d = 1$ and $\mathcal{X} = [0, 1]$ is the unit interval. It is not too difficult to show that the n_s point set

$$\mathcal{D} = \left\{ \frac{1}{2n_s}, \frac{3}{2n_s}, \ldots, \frac{2n_s - 1}{2n_s} \right\}$$

has discrepancy $D_\infty(\mathcal{D}) = 1/2n_s$ because $F(x) = x$ in this case. ♦

Another important measure of discrepancy is the L_p discrepancy of $\mathcal{D}$ which is denoted by $D_p(\mathcal{D})$ and defined by

$$D_p(\mathcal{D}) = \left[\int_{\mathcal{X}} \left| F_{n_s}(\boldsymbol{x}) - F(\boldsymbol{x}) \right|^p d\boldsymbol{x} \right]^{1/p} .$$

The L_∞ discrepancy of $\mathcal{D}$ is a limiting case of L_p discrepancy obtained by letting p go to infinity.

Niederreiter (1992) discussed the use of discrepancy for generating uniformly distributed sequences of points by quasi-Monte Carlo methods. Designs taking observations at sets of points with small discrepancies would be considered more uniform or more spread out than designs corresponding to sets with larger discrepancies. *Uniform designs* take observations at a set of points that minimizes D_p.

Other than the fact that it seems intuitively reasonable to use designs that are spread uniformly over $\mathcal{X} = [0, 1]^d$, why might one consider using a uniform design? One reason that has been proposed is the following. Suppose we are interested in estimating the mean of $g(y(\boldsymbol{X}))$,

$$\mu = \mathrm{E}\left[g(y(\boldsymbol{X}))\right] = \int_{\mathcal{X}} g(y(\boldsymbol{x}))\, d\boldsymbol{x} ,$$

where $g(\cdot)$ is some known function. We consider the properties of the naíve moment estimator

$$T = T(y(\boldsymbol{X}_1), \ldots, y(\boldsymbol{X}_{n_s})) = \frac{1}{n_s} \sum_{j=1}^{n_s} g(y(\boldsymbol{X}_j)) .$$

The Koksma–Hlawka inequality (Niederreiter (1992)) gives an upper bound on the absolute error of this estimator, namely,

$$|\, T(y(\boldsymbol{x}_1), \ldots, y(\boldsymbol{x}_{n_s})) - \mu \,| \leq D_\infty(\mathcal{D})\, V(g) ,$$

where $V(g)$ is a measure of the variation of g that does not depend on $\mathcal{D}$ (see page 19 of Niederreiter (1992) for the definition of $V(g)$). For fixed $g(\cdot)$, this bound is a minimum when $\mathcal{D}$ has minimum discrepancy. This suggests that a uniform design may control the maximum absolute error of T as an estimator of μ. Also, because this holds for any $g(\cdot)$, it suggests that uniform designs may be robust to the choice of $g(\cdot)$.

However, just because an upper bound on the absolute error is minimized, it does not necessarily follow that a uniform design minimizes the maximum absolute error over $\mathcal{X}$ or has other desirable properties. Furthermore, in the context of computer experiments, we are usually not interested in estimating μ. Thus, the above is not a completely compelling reason to use a uniform design in computer experiments as discussed here.

Wiens (1991) provided another reason for considering uniform designs. Suppose one believes the response $y(\boldsymbol{x})$ follows the regression model

$$y(\boldsymbol{x}) = \beta_0 + \sum_{i=1}^{k} \beta_i f_i(\boldsymbol{x}) + \varphi(\boldsymbol{x}) + \epsilon\,,$$

where the $\{f_i\}$ are known functions, the β_i unknown regression parameters, φ is an unknown function representing model bias, and ϵ is Gaussian random error. Wiens (1991) shows that under certain conditions on φ, the uniform design is best in the sense of maximizing the power of the overall F test of the regression.

Fang et al. (2000) provided yet another reason why one may wish to use uniform designs. They note that in orthogonal designs, the points are typically uniformly spread out over the design space. Thus, there is the possibility that uniform designs may often be orthogonal. To explore this further, they use computer algorithms to find designs that minimize a variety of measures of discrepancy and in doing so generate a number of orthogonal designs. Efficient algorithms for generating designs that minimize certain measures of discrepancy, therefore, may be useful in searching for orthogonal designs.

Fang et al. (2000) discussed a method for constructing (nearly) uniform designs. In general, finding a uniform design is not easy. One way to simplify the problem is to reduce the domain of $\mathcal{X}$, perhaps to a finite set of candidate points. Obviously, a uniform design over this reduced domain may not be uniform over $\mathcal{X}$, but suitable selection of a reduced domain may yield designs which are nearly uniform.

A related way to simplify the problem is to reduce the set of candidate designs to some large, finite set. For example, one could restrict attention to only LHDs and then select the one with the minimum discrepancy from the uniform distribution.

As previously, for purposes of what follows, assume $\mathcal{X} = [0,1]^d$. Based on the uniform design for $d = 1$ given in Example 5.12, one might proceed as follows. Let $\boldsymbol{\Pi} = (\Pi_{ij})$ be an $n_s \times d$ matrix such that each column of $\boldsymbol{\Pi}$ is a permutation of the integers $\{1, 2, \ldots, n_s\}$. Let $\boldsymbol{X}(\boldsymbol{\Pi}) = (x_{i,j})$ be the $n_s \times d$ matrix defined by

$$x_{i,j} = (\Pi_{ij} - 0.5)/n_s\,,$$

for all i, j. The n_s rows of $\boldsymbol{X}$ define n_s points in $\mathcal{X} = [0,1]^d$. Hence, each matrix $\boldsymbol{\Pi}$ determines an n_s-point design. For example, when $d = 1$, if $\boldsymbol{\Pi} = (1, 2, \ldots, n_s)^\top$, then

$$\boldsymbol{X}(\boldsymbol{\Pi}) = \left(\frac{1}{2n_s}, \frac{3}{2n_s}, \ldots, \frac{2n_s - 1}{2n_s}\right)^\top,$$

which is the uniform design in $d = 1$ dimension. Note that the n_s rows of $\boldsymbol{X}(\boldsymbol{\Pi})$ correspond to the sample points of an LHD with points at the centers of each sampled cell. One might search over the set $\mathcal{P}$ of all possible permutations $\boldsymbol{\Pi}$, selecting the $\boldsymbol{\Pi}$ that produces the n_s-point design with minimum discrepancy. One would hope that this choice of design is nearly uniform over $\mathcal{X}$. Fang et al. (2000) describe two algorithms for conducting such a search. Bratley et al. (1994) is an additional source for an algorithm that can be used to generate low-discrepancy sequences of points and hence (near) uniform designs.

The discrepancies D_∞ for two designs that appear to be equally uniform may not be the same. The following example illustrates such a case.

Example 5.13. Suppose $d = 2$, $\mathcal{X} = [0,1]^2$, and consider the class of all designs generated by the set of permutations $\mathcal{P}$ introduced in the previous paragraph. One member of this class of designs is

$$\mathcal{D}_{diag} = \left\{ \left(\frac{1}{2n_s}, \frac{1}{2n_s} \right), \left(\frac{3}{2n_s}, \frac{3}{2n_s} \right), \ldots, \left(\frac{2n_s - 1}{2n_s}, \frac{2n_s - 1}{2n_s} \right) \right\}.$$

This n_s-point design takes observations along the diagonal extending from the origin to the point $(1,1)$. Intuitively, we would expect $\mathcal{D}_{diag}$ to be a poor design, because it takes observations only along the diagonal and does not spread observations over $[0,1]^2$. To compute the discrepancy of $\mathcal{D}_{diag}$, we first compute the empirical distribution function F_{n_s} for $\mathcal{D}_{diag}$ at an arbitrary point $\boldsymbol{x} = (x_1, x_2)$ in $[0,1]^2$. Notice that points in $\mathcal{D}_{diag}$ have both coordinates equal and it is straightforward to show from (5.6.2) that

$$F_{n_s}(x_1, x_2) = \frac{\text{number of pts. in } \mathcal{D}_{diag} \text{ with first coordinate} \le \min\{x_1, x_2\}}{n_s}.$$

Notice that $F_{n_s}(\cdot,\cdot)$ is constant almost everywhere except for jumps of size $1/n_s$ at points for which one of the coordinates takes one of the values $\frac{1}{2n_s}, \frac{3}{2n_s}, \ldots, \frac{2n_s-1}{2n_s}$. In particular, $F_{n_s}(x_1, x_2)$ has value $\frac{m}{n_s}$ $(1 \le m \le n_s)$ on the set

$$\mathcal{X}_m = \left\{ (x_1, x_2) \in [0,1]^2 : \frac{2m-1}{2n_s} \le \min\{x_1, x_2\} < \frac{2m+1}{2n_s} \right\}.$$

Recall from (5.6.1) that $F(\cdot)$ is the uniform distribution,

$$F(\boldsymbol{x}) = x_1 x_2$$

on $\mathcal{X} = [0,1]^2$. On $\mathcal{X}_m$, the minimum value of $F(\boldsymbol{x})$ is $\left(\frac{2m-1}{2n_s}\right)^2$ and the supremum of $F(\boldsymbol{x})$ is $\frac{2m+1}{2n_s}$. This supremum is obtained in the limit as $\epsilon \to 0$ along the sequence of points $\left(\frac{2m+1}{2n_s} - \epsilon, 1\right)$. Thus, over $\mathcal{X}_m$, the supremum of $\left|F_{n_s}(\boldsymbol{x}) - F(\boldsymbol{x})\right|$ is either $\left|\frac{m}{n_s} - \left(\frac{2m-1}{2n_s}\right)^2\right|$ or $\left|\frac{m}{n_s} - \frac{2m+1}{2n_s}\right| = \frac{1}{2n_s}$. For $1 \le m \le n_s$, it is not difficult to show that

$$\left| \frac{m}{n_s} - \left(\frac{2m-1}{2n_s} \right)^2 \right| > \frac{1}{2n_s}.$$

Hence, over the set of all points $\boldsymbol{x}$ for which $F_{n_s}(\boldsymbol{x})$ has value $\frac{m}{n_s}$, the supremum of $\left|F_{n_s}(\boldsymbol{x}) - F(\boldsymbol{x})\right|$ is

$$\frac{m}{n_s} - \left(\frac{2m-1}{2n_s} \right)^2 = \frac{n_s m - m^2 + m}{n_s^2} - \frac{1}{4n_s^2},$$

and this occurs at the point $(\frac{2m-1}{2n_s}, \frac{2m-1}{2n_s}) \in \mathcal{D}_{diag}$. Using calculus, one can show that the value of m that maximizes $\frac{n_s m - m^2 + m}{n_s^2} - \frac{1}{4n_s^2}$ is $\frac{n_s+1}{2}$ if n_s is odd, and $\frac{n_s}{2}$ if n_s is even. If n_s is odd, one obtains

$$D_\infty(\mathcal{D}_{diag}) = \sup_{x \in \mathcal{X}} \left|F_{n_s}(\boldsymbol{x}) - F(\boldsymbol{x})\right| = \frac{1}{4} + \frac{1}{2n_s}$$

and if n_s is even,

$$D_\infty(\mathcal{D}_{diag}) = \frac{1}{4} + \frac{1}{2n_s} - \frac{1}{4n_s^2}.$$

However, notice that when n_s is odd, *any* design corresponding to a permutation in $\mathcal{P}$ taking $\frac{n_s+1}{2}$ of its observations at points which are less than or equal to $(1/2, 1/2)$ (under componentwise ordering of vectors) will have support on a set with a discrepancy that is greater than or equal to that of $\mathcal{D}_{diag}$. To see this, simply notice this discrepancy must be at least equal to the value of $\left|F_{n_s}(\boldsymbol{x}) - F(\boldsymbol{x})\right|$ at $\boldsymbol{x} = (1/2, 1/2)$, which is equal to $D_\infty(\mathcal{D}_{diag})$. Likewise, if n_s is even, *any* design taking half of its observations at points less than or equal to $\left(\frac{n_s-1}{2n_s}, \frac{n_s-1}{2n_s}\right)$ will have support on a set with a discrepancy that is greater than or equal to that of $\mathcal{D}_{diag}$. Thus, $\mathcal{D}_{diag}$ is more uniform than any such design, even if such a design spreads points more evenly over $[0, 1]^2$ than simply placing them along the diagonal.

Now consider the n_s-point design,

$$\mathcal{D}_{antidiag} = \left\{ \left(\frac{1}{2n_s}, \frac{2n_s-1}{2n_s} \right), \left(\frac{3}{2n_s}, \frac{2n_s-3}{2n_s} \right), \ldots, \left(\frac{2n_s-1}{2n_s}, \frac{1}{2n_s} \right) \right\}.$$

This design takes observations along the antidiagonal that runs from the point $(0, 1)$ to the point $(1, 0)$. For this design, we notice that when n_s is odd, $F_{n_s}(\boldsymbol{x}) = 0$ at $\boldsymbol{x} = \left(\frac{1}{2} - \epsilon, \frac{n_s+2}{2n_s} - \epsilon\right)$ and so, at this $\boldsymbol{x}$,

$$\left|F_{n_s}(\boldsymbol{x}) - F(\boldsymbol{x})\right| = \left(\frac{1}{2} - \epsilon \right) \left(\frac{n_s+2}{2n_s} - \epsilon \right).$$

In the limit as $\epsilon \to 0$,

$$\left|F_{n_s}(\boldsymbol{x}) - F(\boldsymbol{x})\right| \to \frac{1}{4} + \frac{1}{2n_s}.$$

One can show that this is, in fact, the supremum value of $\left|F_{n_s}(\boldsymbol{x}) - F(\boldsymbol{x})\right|$ for $\mathcal{D}_{antidiag}$, hence its discrepancy is $D_\infty(\mathcal{D}_{antidiag}) = \frac{1}{4} + \frac{1}{2n_s}$. Notice that $\frac{1}{4} + \frac{1}{2n_s}$ is also the value of $D_\infty(\mathcal{D}_{diag})$, so D_∞ considers $\mathcal{D}_{diag}$ and $\mathcal{D}_{antidiag}$ equally uniform when n_s is odd.

When n_s is even, by considering the point $\boldsymbol{x} = \left(\frac{n_s+1}{2n_s} - \epsilon, \frac{n_s+1}{2n_s} - \epsilon\right)$, one can show that in the limit as $\epsilon \to 0$,

$$\left|F_{n_s}(\boldsymbol{x}) - F(\boldsymbol{x})\right| \to \frac{1}{4} + \frac{1}{2n_s} + \frac{1}{4n_s^2}\,.$$

In this case, $D_\infty(\mathcal{D}_{antidiag})$ is at least as large as $\frac{1}{4} + \frac{1}{2n_s} + \frac{1}{4n_s^2}$. Notice that this quantity is larger than the discrepancy of $\mathcal{D}_{diag}$ when n_s is even, so in this case $\mathcal{D}_{diag}$ is a more uniform design than $\mathcal{D}_{antidiag}$. Most readers would consider both designs to be equally uniform. ♦

This example shows that discrepancy, at least as measured by D_∞, may not adequately reflect our intuitive notion of what it means for points to be evenly spread over $\mathcal{X}$. Other measures of discrepancy may perform better. In view of Wiens (1991), uniform designs may be promising, but additional study of their properties in the context of computer experiments is needed. In addition, it is not clear to what extent our intuitive notions of points being evenly spread over $\mathcal{X}$ correspond to objective measures of the performance of a design.

It should be noted that in Fang et al. (2000), the design $\mathcal{D}_{diag}$ is eliminated from consideration because only matrices $\boldsymbol{\Pi}$ of rank d are considered, and the matrix $\boldsymbol{\Pi}$ corresponding to $\mathcal{D}_{diag}$ is of rank 1. Fang et al. (2006) includes an extensive discussion of uniform designs. Also see the list of software in Sect. 5.7.4 for constructing uniform designs.

5.7 Chapter Notes

5.7.1 *Proof That T_L is Unbiased and of the Second Part of Theorem 5.1*

The notation of Sect. 5.2.3 carries over to this subsection. Computation of $E[T_L]$ is facilitated by the following description of how the LH sample is constructed. For each i, divide the range $[0, 1]$ of the i^{th} coordinate of $\boldsymbol{X}$ into n_s intervals of equal marginal probability $\frac{1}{n_s}$ under F. Sample once from each of these intervals and let these sample values be denoted $X_{i1}, X_{i2}, \ldots, X_{in_s}$. Form the $d \times n_s$ array

$$\begin{pmatrix} X_{11} & X_{12} & \ldots & X_{1n_s} \\ X_{21} & X_{22} & \ldots & X_{2n_s} \\ & & \vdots & \\ X_{d1} & X_{d2} & \ldots & X_{dn_s} \end{pmatrix}$$

and then randomly permute the elements in each row using independent permutations. The n_s columns of the resulting array are the LH sample. This is essentially the procedure for selecting an LH sample that was discussed in Sect. 5.2.2. Another way to select an LH sample is as follows. The Cartesian product of the d subintervals [0, 1] partitions $\mathcal{X}$ into n_s^d cells, each of probability $1/n_s^d$. Each of these n_s^d cells can be labeled by a set of d coordinates

$$\boldsymbol{m}_i = (m_{i1}, m_{i2}, \ldots, m_{id}),$$

where $1 \leq i \leq n_s^d$ and m_{ij} is a number between 1 and n_s corresponding to which of the n_s intervals of [0, 1] is represented in cell i. For example, suppose $n_s = 3$, $d = 2$, and $F(\cdot)$ is uniform. The interval [0, 1] is divided into the three intervals $[0, \frac{1}{3})$, $[\frac{1}{3}, \frac{2}{3})$, and $[\frac{2}{3}, 1]$. Similarly for $[a_2, b_2]$. In this case the cell $[\frac{1}{3}, \frac{2}{3}) \times [\frac{1}{3}, \frac{2}{3})$ would have cell coordinates (2, 2).

To obtain an LH sample, select a random sample of n_s of the n_s^d cells, say $\boldsymbol{m}_{i_1}, \boldsymbol{m}_{i_2}, \ldots, \boldsymbol{m}_{i_{n_s}}$, subject to the condition that for each j, the set $\{m_{i_\ell j}\}_{\ell=1}^{n_s}$ is a permutation of the integers $1, 2, \ldots, n_s$. A single point from each of these n_s cells is then randomly selected. For an LH sample obtained in this manner, the density of $\boldsymbol{X}$, given $\boldsymbol{X} \in$ cell i, is

$$f(\boldsymbol{x} \mid \boldsymbol{X} \in \text{cell } i) = \begin{cases} n_s^d\, f(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in \text{cell } i \\ 0 & \text{otherwise} \end{cases} .$$

Thus, the distribution of the output $y(\boldsymbol{X})$ under LH sampling is

$$\begin{aligned} P\left[y(\boldsymbol{X}) \leq y\right] &= \sum_{i=1}^{n_s^d} P\left[y(\boldsymbol{X}) \leq y \mid \boldsymbol{X} \in \text{cell } i\right] P\left[\boldsymbol{X} \in \text{cell } i\right] \\ &= \sum_{i=1}^{n_s^d} \int_{\text{cell } i \text{ and } y(\boldsymbol{x}) \leq y} n_s^d\, f(\boldsymbol{x}) \left(\frac{1}{n_s^d}\right) d\boldsymbol{x} \\ &= \int_{y(\boldsymbol{x}) \leq y} f(\boldsymbol{x})\, d\boldsymbol{x}\,, \end{aligned}$$

which is the same as for random sampling. Hence $E[T_L] = \mu$.

To compute $Var[T_L]$, the sampling procedure is viewed as follows. First the $\boldsymbol{X}_i$ are selected independently and randomly according to the distribution of F from each of the n_s^d cells. Next the sample of n_s cells is selected independently as described above, letting

$$W_i = \begin{cases} 1 \text{ if cell } i \text{ is in the sample} \\ 0 \text{ otherwise} \end{cases}$$

and

$$G_i = g(y(\boldsymbol{X}_i)).$$

Then

$$\begin{aligned} Var\,[T_L] &= Var\left[\frac{1}{n_s}\sum_{j=1}^{n_s} G_j\right] \\ &= \frac{1}{n_s^2}\left[\sum_{i=1}^{n_s^d} Var\,[W_i\,G_i] \right. \\ &\quad \left. + \sum_{i=1}^{n_s^d}\sum_{j=1, j\neq i}^{n_s^d} Cov\left[W_i\,G_i, W_j\,G_j\right]\right]. \end{aligned}$$

Some additional properties of the W_i must be invoked to compute the variances and covariances on the right-hand side of this expression. These results follow from the fundamental rule that the probability of an event is the proportion of samples in which the event occurs. First, $P[W_i = 1] = n_s/n_s^d = 1/n_s^{d-1}$, so W_i is Bernoulli with probability of success $1/n_s^{d-1}$. Second, if W_i and W_j correspond to cells having at least one common cell coordinate, then these two cells cannot both be selected, hence $E[W_i\,W_j] = 0$. Third, if W_i and W_j correspond to cells having no cell coordinates in common, then

$$E\left[W_i\,W_j\right] = P\left[W_i = 1, W_j = 1\right] = \frac{1}{n_s^{d-1}(n_s-1)^{d-1}}.$$

This follows from the fact that, taking order into account, there are $n_s^d(n_s - 1)^d$ pairs of cells with no coordinates in common, and in a sample of size n_s, there are $n_s(n_s - 1)$ such pairs.

Using the fact that for two random variables Z and V, $Var[Z] = E[Var[Z \mid V]] + Var[E[Z \mid V]]$,

$$\begin{aligned} Var\,[W_i\,G_i] &= E\,[Var\,[W_i\,G_i \mid W_i]] + Var\,[E\,[W_i\,G_i \mid W_i]] \\ &= E\left[W_i^2\,Var\,[G_i \mid W_i]\right] + Var\,[W_i\,E\,[G_i \mid W_i]] \\ &= E\left[W_i^2\,Var\,[G_i]\right] + Var\,[W_i\,E\,[G_i]] \qquad (5.7.1) \\ &= E\left[W_i^2\right] Var\,[G_i] + E^2\,[G_i]\,Var\,[W_i], \end{aligned}$$

where the fact that $\boldsymbol{X}_i$ (and hence G_i) and W_i are independent is used in (5.7.1) above. Letting

$$\mu_i = E[g(y(\boldsymbol{X}_i))] = E\left[g(y(\boldsymbol{X})) \mid \boldsymbol{X} \in \text{cell } i\right]$$

and recalling that W_i is Bernoulli,

$$\begin{aligned}\sum_{i=1}^{n_s^d} Var\left[W_i G_i\right] &= \sum_{i=1}^{n_s^d} \left\{E\left[W_i^2\right] Var\left[G_i\right] + E^2\left[G_i\right] Var\left[W_i\right]\right\} \\ &= \frac{1}{n_s^{d-1}} \sum_{i=1}^{n_s^d} \left\{E\left[(G_i - \mu_i)^2\right] + \left(1 - \frac{1}{n_s^{d-1}}\right)\mu_i^2\right\} \\ &= \frac{1}{n_s^{d-1}} \sum_{i=1}^{n_s^d} \left\{\int_{\text{cell } i} (g(y(\boldsymbol{x})) - \mu + \mu - \mu_i)^2 \; n_s^d f(\boldsymbol{x})\, d\boldsymbol{x}\right. \\ &\quad \left. + \left(1 - \frac{1}{n_s^{d-1}}\right)\mu_i^2\right\} \\ &= n_s\, Var\left[g(y(\boldsymbol{X}))\right] - \frac{1}{n_s^{d-1}} \sum_{i=1}^{n_s^d} \left\{(\mu - \mu_i)^2 - \left(1 - \frac{1}{n_s^{d-1}}\right)\mu_i^2\right\}.\end{aligned}$$

Because W_ℓ and $G_\ell = g(y(\boldsymbol{X}_\ell))$ are independent, then for $i \neq j$,

$$\begin{aligned}Cov\left[W_i G_i, W_j G_j\right] &= E\left[W_i G_i W_j G_j\right] - E\left[W_i G_i\right] E\left[W_j G_j\right] \\ &= E\left[W_i W_j\right] E\left[G_i G_j\right] - E\left[W_i\right] E\left[G_i\right] E\left[W_j\right] E\left[G_j\right] \\ &= E\left[W_i W_j\right] E\left[G_i\right] E\left[G_j\right] - \frac{1}{n_s^{d-1}} E\left[G_i\right] \frac{1}{n_s^{d-1}} E\left[G_j\right] \\ &= E\left[W_i W_j\right] \mu_i \mu_j - \frac{1}{n_s^{2d-2}} \mu_i \mu_j .\end{aligned}$$

Hence

$$\sum_{i=1}^{n_s^d} \sum_{j=1, j\neq i}^{n_s^d} Cov\left[W_i G_i, W_j G_j\right] = \sum_{i=1}^{n_s^d} \sum_{j=1, j\neq i}^{n_s^d} \left\{E\left[W_i W_j\right] \mu_i \mu_j - \frac{1}{n_s^{2d-2}} \mu_i \mu_j\right\}.$$

Recall that $E[W_i W_j] = 0$ if cells i and j have at least one common cell coordinate. Let R denote the $n_s^d (n_s - 1)^d$ pairs of cells (with regards to order) having no cell coordinates in common. On this set,

$$E\left[W_i W_j\right] = \frac{1}{n_s^{d-1} (n_s - 1)^{d-1}} ,$$

giving

$$Var\left[\frac{1}{n_s} \sum_{j=1}^{n_s} G_j\right] = \frac{1}{n_s^2} \left\{n_s\, Var\left[g(y(\boldsymbol{X}))\right] - \frac{1}{n_s^{d-1}} \sum_{i=1}^{n_s^d} (\mu - \mu_i)^2\right.$$

$$+\frac{1}{n_s^{d-1}}\left(1-\frac{1}{n_s^{d-1}}\right)\sum_{i=1}^{n_s^d}\mu_i^2$$

$$\left. +\frac{1}{n_s^{d-1}(n_s-1)^{d-1}}\sum_R \mu_i\,\mu_j - \frac{1}{n_s^{2d-2}}\sum_{i=1}^{n_s^d}\sum_{j=1,j\neq i}^{n_s^d}\mu_i\,\mu_j\right\}.$$

Notice that

$$\begin{aligned}\sum_{i=1}^{n_s^d}\mu_i &= \sum_{i=1}^{n_s^d} E\left[g(y(\boldsymbol{X}))\mid \boldsymbol{X}\in \text{cell } i\right]\\ &= \sum_{i=1}^{n_s^d}\int_{\text{cell } i} g(y(\boldsymbol{x}))\, n_s^d\, f(\boldsymbol{x})\,d\boldsymbol{x}\\ &= n_s^d\int_{\mathcal{X}} g(y(\boldsymbol{x}))\,f(\boldsymbol{x})\,d\boldsymbol{x} = n_s^d\,\mu\,.\end{aligned}$$

So

$$\begin{aligned}Var\left[\frac{1}{n_s}\sum_{j=1}^{n_s}G_j\right] &= \frac{1}{n_s}Var\left[g(y(\boldsymbol{X}))\right] - \frac{1}{n_s^{d+1}}\sum_{i=1}^{n_s^d}\left(\mu^2 - 2\mu_i\,\mu + \mu_i^2\right)\\ &\quad + \left(\frac{1}{n_s^{d+1}}-\frac{1}{n_s^{2d}}\right)\sum_{i=1}^{n_s^d}\mu_i^2\\ &\quad + \frac{1}{n_s^{d+1}(n_s-1)^{d-1}}\sum_R\mu_i\,\mu_j - \frac{1}{n_s^{2d}}\sum_{i=1}^{n_s^d}\sum_{j=1,j\neq i}^{n_s^d}\mu_i\,\mu_j\\ &= Var\left[T_R\right] + \frac{1}{n_s}\mu^2 - \frac{1}{n_s^{2d}}\left(\sum_{i=1}^{n_s^d}\mu_i\right)^2\\ &\quad + \frac{1}{n_s^{d+1}(n_s-1)^{d-1}}\sum_R\mu_i\,\mu_j\\ &= Var\left[T_R\right] - \frac{n_s-1}{n_s}\mu^2\\ &\quad + \left(\frac{n_s-1}{n_s}\right)\left(\frac{1}{n_s^d(n_s-1)^d}\right)\left(\sum_R\mu_i\,\mu_j\right)\\ &= Var\left[T_R\right]\end{aligned}$$

$$
\begin{aligned}
&-\left(\frac{n_s-1}{n_s}\right)\left(\frac{1}{n_s^d(n_s-1)^d}\right)\left(\sum_R \mu^2\right) \\
&+\left(\frac{n_s-1}{n_s}\right)\left(\frac{1}{n_s^d(n_s-1)^d}\right)\left(\sum_R \mu_i\mu_j\right) \\
= {} & Var\,[T_R] \\
&+\left(\frac{n_s-1}{n_s}\right)\left(\frac{1}{n_s^d(n_s-1)^d}\right) \\
&\times \sum_R (\mu_i-\mu)(\mu_j-\mu) \qquad (5.7.2) \\
\leq {} & Var\,[T_R]\,,
\end{aligned}
$$

provided the last term in (5.7.2) is less than or equal to 0. Thus, whether LH sampling is superior to simple random sampling depends on the sign of this term, which in turn depends on the nature of g and f. Note also that LH sampling is superior to stratified random sampling with proportional sampling if

$$\left(\frac{n_s-1}{n_s}\right)\left(\frac{1}{n_s^d(n_s-1)^d}\right)\sum_R (\mu_i-\mu)(\mu_j-\mu) \; < -\frac{1}{n_s}\sum_{i=1}^{I} p_i(\mu-\tau_i)^2\,,$$

where the input space has been partitioned into I disjoint strata $\{S_i\}$, $p_i = P[\boldsymbol{X} \in S_i]$ is the probability of sampling from S_i, and $\tau_i = E[g(y(\boldsymbol{X})) \mid \boldsymbol{X} \in S_i]$ is the i^{th} stratum mean. McKay et al. (1979) prove that under the assumptions of Theorem 5.1, if $y(x_1,\ldots,x_d)$ is monotonic in each of its arguments and $g(w)$ is a monotonic function of w, then $\sum_R(\mu_i-\mu)(\mu_j-\mu) \leq 0$. This completes the proof of Theorem 5.1. ♦

5.7.2 The Use of LHDs in a Regression Setting

Owen (1992b) presents a multivariate extension of Theorem 5.3 and its application to computer experiments when fitting a regression to output data (rather than the constant mean described in Sect. 5.2.3). The basis for the application is the following multivariate version of Theorem 5.3. The setting is as follows. Suppose that $\boldsymbol{X}$ has independent components with distribution function $F(\cdot)$, $\boldsymbol{y}(\boldsymbol{X}) = (y_1(\boldsymbol{X}),\ldots,y_k(\boldsymbol{X}))^\top$, $\overline{\boldsymbol{Y}} = \frac{1}{n_s}\sum_{i=1}^{n_s}\boldsymbol{y}(\boldsymbol{X}_i)$ and $\boldsymbol{\mu} = \int_{\mathcal{X}}\boldsymbol{y}(\boldsymbol{x})\,d\boldsymbol{x}$.

Corollary 5.1. Let $r_\ell(\boldsymbol{x})$ be the residual from additivity for $y_\ell(\boldsymbol{x})$ (see the discussion preceding Theorem 5.2 for the definition of the residual from additivity) and define

$$\sigma_{ij} = \int_{\mathcal{X}} r_i(\boldsymbol{x})\,r_j(\boldsymbol{x})\,dF(\boldsymbol{x}).$$

Let $\boldsymbol{\Sigma}$ be the $d \times d$ matrix whose (i, j) entry is σ_{ij}. Then $\sqrt{n_s}\,(\overline{\boldsymbol{Y}} - \boldsymbol{\mu})$ tends in distribution to $N_k(\boldsymbol{0}, \boldsymbol{\Sigma})$ as $n_s \to \infty$.

Let $\boldsymbol{Z}(\boldsymbol{x})$ be a vector-valued function for which a linear model $\boldsymbol{Z}(\boldsymbol{x})\boldsymbol{\beta}$ is an appropriate approximation to $\boldsymbol{Y}(\boldsymbol{x})$. The "population" least squares value of $\boldsymbol{\beta}$ is

$$\boldsymbol{\beta}_{\text{POP}} \equiv \left[\int_{\mathcal{X}} \boldsymbol{Z}^\top(\boldsymbol{x})\boldsymbol{Z}(\boldsymbol{x})\, dF(\boldsymbol{x})\right]^{-1} \int_{\mathcal{X}} \boldsymbol{Z}^\top(\boldsymbol{x})\boldsymbol{Y}(\boldsymbol{x})\, dF(\boldsymbol{x}).$$

Assuming $\int_{\mathcal{X}} \boldsymbol{Z}^\top(\boldsymbol{x})\boldsymbol{Z}(\boldsymbol{x})\, dF(\boldsymbol{x})$ is known or easily computable (e.g., this would be the case for polynomial regression), $\boldsymbol{\beta}_{\text{POP}}$ is estimated by

$$\widehat{\boldsymbol{\beta}}_{\text{POP}} = \left[\int_{\mathcal{X}} \boldsymbol{Z}^\top(\boldsymbol{x})\boldsymbol{Z}(\boldsymbol{x})\, dF(\boldsymbol{x})\right]^{-1} \frac{1}{n_s}\sum_{i=1}^{n_s} \boldsymbol{Z}^\top(\boldsymbol{X}_i)\boldsymbol{Y}(\boldsymbol{X}_i).$$

The variance of $\widehat{\boldsymbol{\beta}}_{\text{POP}}$ is of the "sandwich" form

$$\left[\int_{\mathcal{X}} \boldsymbol{Z}^\top(\boldsymbol{x})\boldsymbol{Z}(\boldsymbol{x})\, dF(\boldsymbol{x})\right]^{-1} \boldsymbol{\Sigma} \left[\int_{\mathcal{X}} \boldsymbol{Z}^\top(\boldsymbol{x})\boldsymbol{Z}(\boldsymbol{x})\, dF(\boldsymbol{x})\right]^{-1},$$

where $\boldsymbol{\Sigma}$ is defined in Corollary 5.1 above using the j^{th} element of $\boldsymbol{Z}^\top(\boldsymbol{x})\boldsymbol{Y}(\boldsymbol{x})$ in place of $Y_j(\boldsymbol{x})$ in the definition of $r_j(\boldsymbol{x})$. Appealing to Theorem 5.2, one might argue to the extent that $\boldsymbol{Z}^\top(\boldsymbol{x})\boldsymbol{Y}(\boldsymbol{x})$ is additive, the regression may be more accurately estimated from a LHD than from a design based on a simple random sample.

Owen (1992b) discusses some other estimators of $\boldsymbol{\beta}_{\text{POP}}$. The point is that when a linear model is likely to provide a good approximation to $\boldsymbol{y}(\boldsymbol{x})$, using a LHD followed by regression modeling is not an unreasonable approach to predicting simulator output.

5.7.3 Other Space-Filling Designs

The methods discussed in this chapter are not the only ones that generate space-filling designs. The literature on numerical integration contains numerous suggestions for constructing evenly spaced designs. Niederreiter (1992) contains a wealth of information about such designs, including their mathematical properties.

As mentioned in Sect. 5.2.1, one possibility is to choose points on a regularly spaced grid superimposed on the experimental region. For example, if the experimental region is $\mathcal{X} = [0, 1]^d$, the d-fold Cartesian product of the n_s point set

$$S = \left\{\frac{1}{2n_s}, \frac{3}{2n_s}, \ldots, \frac{2n_s - 1}{2n_s}\right\}$$

would be a grid consisting of n_s^d points. Grid designs consist of an array of evenly spaced points, but projections onto subspaces have many replicated points.

An improvement over grids is obtained by the method of good lattice points. Such designs are appealing in that they appear evenly spaced and in some cases have attractive properties in numerical integration. Niederreiter (1992) discusses these designs in more detail. Bates et al. (1996) consider lattice designs in the context of computer experiments.

Nets form another class of designs that appear space-filling and which are popular in numerical integration. See Niederreiter (1992) and Owen (1995) for more details.

Because these designs are intended for use in numerical integration, they are generally used in situations where a large sample size is employed. Their properties tend to be for large numbers of observations and their small-sample behavior is not clear (and thus their usefulness in computer experiments in which the total number of observations is constrained to be small).

5.7.4 Software for Constructing Space-Filling Designs

The designs discussed in this chapter must be produced numerically. Below is a partial list of software that will generate these designs.

1. Software for generating *randomly selected Latin Hypercube Designs* is available in both R and MATLAB. The R package DiceDesign will generate random LHD designs, (nearly) maximin LHD designs, and LHD designs with low discrepancy. It will also generate maximin designs.
2. The stand-alone program ACED can generate maximin designs within the class of LHD designs.
3. JMP will generate LHD and maximin designs for rectangular, user-specified design regions and for any (n_s, d). For LHD designs, JMP chooses points so that the design is (nearly) maximin subject to a constraint that maintains even spacing between factor levels. JMP refers to maximin designs as *sphere-packing designs*. JMP will generate two types of designs with good distance-based properties for rectangular and for non-rectangular regions. One type of design is called a "fast flexible filling design." Designs can be generated for regions determined by user-specified linear constraints on the inputs. The algorithm produces designs with good properties using the minimax distance criterion or the MaxPro criterion. The other design type is designated "minimum potential designs." Designs can be generated for spherical regions for any (n_s, d). To generate any of these designs, run the Space Filling Design command under the DOE menu.
4. Dakota is a software package developed at Sandia National Laboratories for the analysis of data from predictive simulations. This package will generate several types of space-filling designs including orthogonal array designs, LHDs, and orthogonal array-based LHDs. Dakota can be downloaded from http://dakota.sandia.gov

5. The DiceDesign package in R will generate uniform designs under a variety of discrepancies including the L_2 discrepancy, the centered L_2 discrepancy, and the star discrepancy.
6. The R package SLHD will generate optimal *sliced LHDs*. See Ba et al. (2015) for a discussion of optimal sliced LHDs, the description of an algorithm to construct such designs, and earlier references to this class of designs.
7. The R package MaxPro generates maximum projection designs either unrestricted, within the class of LHDs, or to augment a given design using the MaxPro criterion. See Joseph et al. (2015) for a discussion of maximum projection designs and the construction of such designs.
8. The stand-alone C code of Brian Williams, available at http://go.osu.edu/LHDesigns, will generate optimal designs for the following scenarios:

   (a) Given the set of dimensions $J \subset \{1, \ldots, d\}$ over which projections are of interest, the C code oalhs will construct a design that maximizes the minimum (normalized) interpoint distance (5.4.8) in the class of OA-based LHDs based on a given starting OA. It can also construct a design that maximizes the average reciprocal distance (5.4.9) for the same class of OA-based LHDs. The website http://neilsloane.com/oadir, maintained by Neal Sloane, contains a comprehensive library of known OAs. These designs are also available at http://go.osu.edu/OrthogonalArrays.
   (b) Given the set of dimensions $J \subset \{1, \ldots, d\}$ over which projections are of interest, the C code slhs will construct a design that maximizes the minimum (normalized) interpoint distance (5.4.8) in the class of symmetric LHDs (see Sect. 5.3.3) of a given number of runs. It can also construct a design that maximizes the average reciprocal distance (5.4.9) for the same class of symmetric LHDs

9. The R program concad computes optimum distance designs of a given number of runs for bounded polygonal input regions, i.e., regions of points $\boldsymbol{x}$ that satisfy $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$ for given $\boldsymbol{A}$ and $\boldsymbol{b}$. The concad package can be used with a maximin or minimum ARD criteria or a compromise criterion involving both maximin and minimum ARD optimality. Contact D. Draguljić (ddraguljic@gmail.com) to obtain the concad package.
10. The R function runif.sobol will generate Sobol´ sequences and the function runif.halton will generate Halton sequences. (See the R documentation for the details of the bases used in the construction and the required R packages.)
11. The MATLAB function haltonset can generate Halton sequences, while the sobolset function will construct Sobol´ sequences. Both functions require the Statistics toolbox. One can also find online MATLAB code for generating Niederreiter sequences. For example, see http://people.sc.fsu.edu/~jburkardt/m_src/niederreiter2/niederreiter2.m.
12. The open-source GSL library contains code for generating Sobol´ and Niederreiter sequences which can be accessed using an appropriate calling routine in C.

13. JMP will generate uniform designs. JMP uses the centered L_2 discrepancy measure of Hickernell (1998). To generate uniform designs, one must run the Space-Filling Design command under the DOE menu.
14. The R package minimaxdesign generates minimax designs and minimax projection designs (using clustering and the particle swarm optimization (PSO) algorithms).

5.7.5 Online Catalogs of Designs

The Mathematics Department of Hong Kong Baptist University maintains the website http://www.math.hkbu.edu.hk/UniformDesign with information about uniform designs, including lists of publications about uniform designs and tables of uniform designs. In addition there are websites that provide catalogs of designs that are optimal under *distance or other criteria*. Of these, http://www.spacefillingdesigns.nl is particularly rich.

Chapter 6
Some Criterion-Based Experimental Designs

6.1 Introduction

Chapter 5 considered designs that attempt to spread observations "evenly" throughout the experimental region. Such designs were called space-filling designs. Recall that one rationale for using a space-filling design is the following. If it is believed that interesting features of the true model are just as likely to be in one part of the input region as another, observations should be taken in all portions of the input region. There are many heuristic criteria for producing designs that might be considered space-filling; several of these were discussed in Chap. 5. However none of the methods was tied to a statistical justification, and no single criterion was singled out as best.

Physical experiments have a long history for choosing designs to satisfy statistical criteria. An example are designs selected to "minimize" the variances of estimators of the mean model parameters. Specifically, suppose it is known that a second-order response surface mean model adequately approximates the output of a physical experiment. The design for an experiment satisfying such a mean model might be selected according to one of the many criteria proposed for second-order response surfaces. One such criterion is *D-optimality*; a D-optimal design minimizes the determinant of the covariance matrix of the least squares estimators of the regression parameters. This is equivalent to minimizing the volume of the confidence ellipsoid for the regression parameters. Many other criteria have been proposed in the statistical literature that are tailored to answering specific inference questions. For example, a researcher may be interested in minimizing the integrated mean squared prediction error or minimizing the squared bias of model predictions (with respect to some true model).

This chapter considers some statistical criteria that have been used to construct experimental designs for *computer experiments*. Constructing designs according to such criteria is often more difficult than in linear model settings because these criteria are functions of the unknown parameters of the Gaussian process models. Analytic results are difficult to obtain and have been found in only a few special cases.

The details of such results are very technical and beyond the scope of this book. Here the criteria are described, and methods are identified for finding good designs according to these criteria. This is an active area of research.

6.2 Designs Based on Entropy and Mean Squared Prediction Error Criterion

6.2.1 Maximum Entropy Designs

To introduce the notion of entropy, let X be a random variable taking a finite number of values. For simplicity, take these values to be $1, 2, \ldots, n$. Let p_i be the probability that $X = i$. Define the *entropy* of X to be

$$H(X) = -\sum_{i=1}^{n} p_i \times \ell n\,(p_i)\,, \tag{6.2.1}$$

where $\ell n(\cdot)$ denotes the natural logarithm and $p \times \ell n\,(p)$ is defined to be 0 when $p = 0$. This definition can be extended to a continuous random variable X having probability density function $f(\cdot)$ by defining the entropy of X to be

$$H(X) = -\int_{\mathcal{X}} f(x) \times \ell n\,(f(x))\,dx$$

where, again, $f(x) \times \ell n\,(f(x))$ is defined to be 0 whenever $f(x) = 0$.

What does entropy represent and why is this particular function used to measure entropy? Entropy is intended to be a measure of the unpredictability of a random variable. Intuitively, if all outcomes of a random variable X are equally likely, i.e., $p_1 = p_2 = \cdots = p_n = 1/n$, then X is maximally unpredictable, and a reasonable definition of entropy should assign maximum value to this distribution. If X takes on a single value with probability 1, then X is completely predictable and should have minimum entropy. That entropy achieves this intuition is seen by first considering X which is a constant with probability one. Then $H(X)$, defined by (6.2.1), equals 0. This is the smallest possible value of $H(X)$ because $H(X) \geq 0$ for every X since all the p_i are between 0 and 1. Now consider a discrete random variable X with $p_1 = p_2 = \cdots = p_n = 1/n$; it is simple to compute that $H(X) = \ell n(n)$. That this is the largest possible value of $H(X)$ can be established by noting that the function $f(x) = \ell n(x)$ is strictly concave on $x > 0$. Thus if X has n support points with associated (positive) probabilities $\{p_1, p_2, \ldots, p_n\}$, then $H(X) = \sum_{i=1}^{n} p_i \times \ell n\,(1/p_i) < \ell n\left(\sum_{i=1}^{n} p_i/p_i\right) = \ell n\,(n)$. In sum, entropy can be thought of as a measure of the uniformity and dispersion of the distribution of the random variable X.

Lindley (1956) tied entropy to the amount of information contained in an experiment. His argument is based on one originally proposed by Shannon (1948). The basic idea is the following. Consider a statistical model for a measured response that is determined by a parameter $\boldsymbol{\xi}$. For example, in regression, $\boldsymbol{\xi}$ might be the vector of regression parameters. In the Gaussian process models considered in this book, $\boldsymbol{\xi}$ might represent the vector of correlation parameters.

For simplicity, assume the set of possible values of $\boldsymbol{\xi}$, Ξ, is finite. Suppose that knowledge about $\boldsymbol{\xi}$ is specified by a (prior or a posterior) probability mass function that is denoted by $\pi_{\boldsymbol{\xi}}(\cdot)$; similar definitions hold for continuous $\boldsymbol{\xi}$ distributions.

The goal is to measure the "information" about $\boldsymbol{\xi}$, denoted I, that is contained in the distribution $\pi_{\boldsymbol{\xi}}(\cdot)$. Lindley describes I as the amount of information that must be provided to "know" $\boldsymbol{\xi}$. Assuming that I is taken to be real-valued, what intuitive properties should it have? One heuristic property arises from considering the following two-stage description of the distribution of $\boldsymbol{\xi}$. Let Ξ_1 be a nonempty proper subset of Ξ with total probability, P, that is strictly between 0 and 1, i.e.,

$$0 < P \equiv \sum_{\boldsymbol{\xi}^\star \in \Xi_1} \pi_{\boldsymbol{\xi}}(\boldsymbol{\xi}^\star) < 1.$$

In the first stage, consider whether $\boldsymbol{\xi} \in \Xi_1$ or its complement. The distribution that summarizes our knowledge of $\boldsymbol{\xi}$ at this stage is $(P, 1-P)$. This provides, say, amount I_0 of information. The information provided in the second stage is, say, I_1 or I_2 according to whether $\boldsymbol{\xi} \in \Xi_1$ or $\boldsymbol{\xi}$ is in the complement of Ξ_1; I_1 and I_2 are calculated from the following distributions. Given $\boldsymbol{\xi} \in \Xi_1$, the conditional distribution $[\boldsymbol{\xi} \mid \boldsymbol{\xi} \in \Xi_1]$ $(= \pi_{\boldsymbol{\xi}}(\cdot)/P)$ summarizes the knowledge about $\boldsymbol{\xi}$. Given that $\boldsymbol{\xi}$ is in the complement of Ξ_1, the conditional distribution $[\boldsymbol{\xi} \mid \boldsymbol{\xi} \notin \Xi_1]$ $(= \pi_{\boldsymbol{\xi}}(\cdot)/(1-P))$ summarizes the knowledge about $\boldsymbol{\xi}$. Then (as Shannon (1948) argued) one should require that the information provided in the first stage and the average (expected) amount of information provided in the second stage add up to the total information I in $\pi_{\boldsymbol{\xi}}(\cdot)$, namely,

$$I = I_0 + P\, I_1 + (1 - P)\, I_2. \tag{6.2.2}$$

The additivity requirement (6.2.2) is the fundamental postulate for a measure of information I in the distribution $\pi_{\boldsymbol{\xi}}(\cdot)$. Shannon (1948) showed that

$$I = \sum_{\boldsymbol{\xi}^\star} \pi_{\boldsymbol{\xi}}(\boldsymbol{\xi}^\star)\, \ell n\left(\pi_{\boldsymbol{\xi}}(\boldsymbol{\xi}^\star)\right)$$

is the only function having this property, apart from an arbitrary multiplying constant and a mild continuity property. As for entropy, I can be extended to continuous distributions with probability density functions $\pi_{\boldsymbol{\xi}}(\cdot)$ by setting

$$I = \int_{\boldsymbol{\xi}^\star} \pi_{\boldsymbol{\xi}}(\boldsymbol{\xi}^\star)\, \ell n\left(\pi_{\boldsymbol{\xi}}(\boldsymbol{\xi}^\star)\right) d\boldsymbol{\xi}^\star .$$

Information is typically viewed as the negative of a measure of entropy, namely, $I = -H$. The above argument suggests that not only is $H(X)$ a reasonable measure of entropy, but it is the *only* measure of entropy having certain desirable properties.

Shewry and Wynn (1987) used these ideas to develop the notion of maximum entropy sampling when the design space is discrete. Let $\pi_{\boldsymbol{\xi}}(\cdot)$ be the *prior* distribution for $\boldsymbol{\xi}$, the parameter in the statistical model to be estimated. Observe a response at the n_s input sites $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{n_s}\}$ and call $\mathcal{D}$ the *experimental design*. Let $\pi_{\boldsymbol{\xi}|\mathcal{D}}(\cdot)$ denote the posterior distribution of $\boldsymbol{\xi}$ given the observations collected using design $\mathcal{D}$. The amount of information about $\boldsymbol{\xi}$ that is contained in the prior before the experiment is

$$I = \int \pi_{\boldsymbol{\xi}}(\boldsymbol{\xi}^\star)\, \ell n\left(\pi_{\boldsymbol{\xi}}(\boldsymbol{\xi}^\star)\right) d\boldsymbol{\xi}^\star;$$

the amount of information about $\boldsymbol{\xi}$ after the experiment using design $\mathcal{D}$ is

$$I_{\mathcal{D}} = \int \pi_{\boldsymbol{\xi}|\mathcal{D}}(\boldsymbol{\xi}^\star)\, \ell n\left(\pi_{\boldsymbol{\xi}|\mathcal{D}}(\boldsymbol{\xi}^\star)\right) d\boldsymbol{\xi}^\star.$$

Thus $I_{\mathcal{D}} - I$ is the change in information. Shewry and Wynn (1987) evaluated the design $\mathcal{D}$ by its expected change in information. Using the fact that entropy is the negative of information, Shewry and Wynn (1987) showed that the expected change in information is *maximized* by that design which maximizes the entropy of the observed responses at the points in the design. Such a design is called a *maximum entropy design*.

In the case of the Gaussian process models, recall that the training data has the conditional distribution

$$[\boldsymbol{Y}^{n_s} \mid \boldsymbol{\beta}] \sim N_{n_s}\left(\boldsymbol{F}\boldsymbol{\beta}, \sigma_Z^2 \boldsymbol{R}\right),$$

assuming σ_Z^2 and $\boldsymbol{R}$ are known. One can show that a *maximum entropy design* $\mathcal{D}$ *maximizes the determinant of the (unconditional) covariance of the observed responses* $\boldsymbol{Y}^{n_s}$ *at the points in the design*. In particular, assuming the Gaussian prior

$$[\boldsymbol{\beta}] \sim N_p\left(\boldsymbol{b}_0, \tau^2 \boldsymbol{V}_0\right),$$

the determinant of this marginal covariance matrix of $\boldsymbol{Y}^{n_s}$ is

$$\begin{aligned}
\det\left(\sigma_Z^2 \boldsymbol{R} + \tau^2 \boldsymbol{F}\boldsymbol{V}_0\boldsymbol{F}^\top\right) &= \det\left(\begin{bmatrix} \sigma_Z^2\boldsymbol{R} + \tau^2\boldsymbol{F}\boldsymbol{V}_0\boldsymbol{F}^\top & \boldsymbol{F} \\ \boldsymbol{0} & \boldsymbol{I}_p \end{bmatrix}\right) \\
&= \det\left(\begin{bmatrix} \sigma_Z^2\boldsymbol{R} & \boldsymbol{F} \\ -\tau^2\boldsymbol{V}_0\boldsymbol{F}^\top & \boldsymbol{I}_p \end{bmatrix} \times \begin{bmatrix} \boldsymbol{I}_n & \boldsymbol{0} \\ \tau^2\boldsymbol{V}_0\boldsymbol{F}^\top & \boldsymbol{I}_p \end{bmatrix}\right) = \det\left(\begin{bmatrix} \sigma_Z^2\boldsymbol{R} & \boldsymbol{F} \\ -\tau^2\boldsymbol{V}_0\boldsymbol{F}^\top & \boldsymbol{I}_p \end{bmatrix}\right) \\
&= \det\left(\begin{bmatrix} \boldsymbol{I}_n & \boldsymbol{0} \\ \tau^2\boldsymbol{V}_0\boldsymbol{F}^\top(\sigma_Z^2\boldsymbol{R})^{-1} & \boldsymbol{I}_p \end{bmatrix} \times \begin{bmatrix} \sigma_Z^2\boldsymbol{R} & \boldsymbol{F} \\ -\tau^2\boldsymbol{V}_0\boldsymbol{F}^\top & \boldsymbol{I}_p \end{bmatrix}\right) \\
&= \det\left(\begin{bmatrix} \sigma_Z^2\boldsymbol{R} & \boldsymbol{F} \\ \boldsymbol{0} & \tau^2\boldsymbol{V}_0\boldsymbol{F}^\top(\sigma_Z^2\boldsymbol{R})^{-1}\boldsymbol{F} + \boldsymbol{I}_p \end{bmatrix}\right) \\
&= \det\left(\sigma_Z^2\boldsymbol{R}\right)\det\left(\tau^2\boldsymbol{V}_0\boldsymbol{F}^\top(\sigma_Z^2\boldsymbol{R})^{-1}\boldsymbol{F} + \boldsymbol{I}_p\right).
\end{aligned}$$

If the $\boldsymbol{\beta}$ prior assumes these coefficients are known and fixed at $\boldsymbol{b}_0$, i.e., $\tau^2 = 0$, the maximum entropy criterion reduces to

$$\det\left(\sigma_Z^2 \boldsymbol{R}\right). \tag{6.2.3}$$

Shewry and Wynn (1987) used the maximum entropy criterion to find designs for certain spatial models. Currin et al. (1991) and Mitchell and Scott (1987) applied maximum entropy to selecting designs for computer experiments. There are three points that the researcher should notice regarding the latter use. First, maximizing (6.2.3) is equivalent to maximizing det($\boldsymbol{R}$) because σ_Z^2 is independent of the design. Second, $\boldsymbol{R}$ depends on the correlation function $R(\cdot)$ which is generally parametric with unknown parameters. Third, the det($\boldsymbol{R}$) is maximized when $\boldsymbol{R}$ is the identity matrix. The argument showing this last statement is as follows. Observe that $\boldsymbol{R}$ is an $n_s \times n_s$ correlation matrix. All $n_s \times n_s$ correlation matrices have trace equal to n_s, which is the sum of their (necessarily nonnegative) eigenvalues. The determinant of such a nonnegative definite matrix is maximized when the product of the n_s eigenvalues is maximized. Subject to the constraint that the sum is n_s, this occurs when the eigenvalues are all equal to 1. The only such correlation matrix is the identity matrix.

One application where the design can force $\boldsymbol{R}$ to be an identity matrix occurs when the correlation function has compact support, such as the cubic correlation function or the Bohman correlation function. In such a case it may be possible to choose the n_s input sites $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{n_s}\}$ so that all are uncorrelated and hence $\boldsymbol{R}$ is the identity matrix. In this case, $\mathcal{D}$ would be a maximum entropy design. For example, suppose $n_s = 4$, $d = 2$, $\mathcal{X} = [0, 1]^2$ and a separable cubic correlation function with correlation parameters $\psi_1 < 1$ and $\psi_2 < 1$ (see (2.2.12)). A design taking observations at the corners of $[0, 1]^2$ would be a maximum entropy design.

One strategy for applying the maximum entropy principle in the case of arbitrary unknown $R(\cdot)$ is to apply the following two-stage procedure. In the first stage, the researcher uses one of the designs discussed in Chap. 5, e.g., an LHD, and estimates the unknown correlation parameters based on the first-stage data. The second stage treats the estimated parameter values as true and applies the maximum entropy criterion to determine a new design. Another strategy might be to carry out a robustness study among designs that are locally maximum entropy; an example of this approach is described in the next subsection.

Currin et al. (1991) described an algorithm adopted from DETMAX (Mitchell (1974)) for finding a maximum entropy design when the correlation function is known. Johnson et al. (1990) suggested that, in a limiting sense as the correlation tends to 0, maximum entropy designs are maximin distance designs.

Figure 6.1 shows three 20-point maximum entropy designs in $d = 2$ dimensions when $(x_1, x_2) \in [0, 1]^2$. The designs are calculated assuming the Gaussian correlation function

$$R(h_1, h_2) = \prod_{i=1}^{2} \exp(-\xi_i \, h_i^2) \, ,$$

where $\boldsymbol{\xi} = (\xi_1, \xi_2)^\top > 0$ is *known*. The three panels of Fig. 6.1 correspond to $\boldsymbol{\xi} = (1, 1)$ in panel (a), $\boldsymbol{\xi} = (25, 25)$ in panel (b), and $\boldsymbol{\xi} = (1, 25)$ in panel (c). The designs in both (a) and (b) appear space-filling with the caveat that the points in (a) appear to be distributed closer to the boundary than those in (b). The design in panel (c) has projections onto the x_2-axis that are more spread out than the projections onto the x_1-axis. Intuitively, this seems sensible because the larger ξ_2 correlation parameter in the Gaussian correlation function means that the process has more volatile (rougher) realizations in x_2 (see Fig. 2.6). The greater volatility in x_2 requires projections onto the x_2 axis to be more finely spread out. Mahalanobis distance, rather than Euclidean distance, is perhaps a more appropriate metric in terms of assessing "space-fillingness" of a design when the Gaussian correlation function is used.

Figure 6.1 raises some interesting issues. First, if the correlation function is believed to be isotropic (as in panels (a) and (b)), maximum entropy designs may be relatively insensitive to the actual values of the correlation parameters. Second, for small values of the correlation parameters, points in a maximum entropy design are clustered a bit closer to the boundary. Third, if the correlation function is strongly non-isotropic (as in panel (c)), the projections of points of a maximum entropy design onto different coordinates may be quite different. The design itself need not look particularly space-filling. Prior knowledge about the level of activity of the different inputs (via the correlation parameters) is essential for selecting a good design.

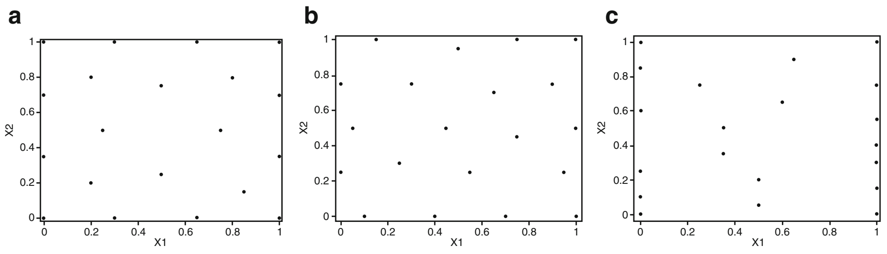


Fig. 6.1 Maximum entropy designs on $[0, 1]^2$ obtained by maximizing (6.2.3). In panel (**a**), $\boldsymbol{\xi} = (1, 1)$; in panel (**b**), $\boldsymbol{\xi} = (25, 25)$; and in panel (**c**), $\boldsymbol{\xi} = (1, 25)$

6.2.2 Mean Squared Prediction Error Designs

Consider the setting of Sect. 3.3 and suppose that the deterministic response $y(\boldsymbol{x})$, $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^d$ is a realization of a stochastic process $Y(\boldsymbol{x})$ according to (2.2.3). Let $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{n_s}\}$ denote an n_s-point design. Assume the BLUP $\widehat{y}(\boldsymbol{x})$ based on $\mathcal{D}$ (see (3.2.7)) is used to predict the response at input $\boldsymbol{x}$.

Sacks et al. (1989a) considered two design criteria, each based on minimizing a functional of the MSPE of $\widehat{y}(\boldsymbol{x})$ which the reader will recall is given by

$$\begin{aligned}\text{MSPE}\left[\widehat{y}(\boldsymbol{x})\right] &= E\left[\left(\widehat{y}(\boldsymbol{x}) - Y(\boldsymbol{x})\right)^2\right] \\ &= \sigma_Z^2\left\{1 - \left(\boldsymbol{f}^\top(\boldsymbol{x})\ \boldsymbol{r}^\top(\boldsymbol{x})\right)\begin{bmatrix}\boldsymbol{0} & \boldsymbol{F}^\top \\ \boldsymbol{F} & \boldsymbol{R}\end{bmatrix}^{-1}\begin{pmatrix}\boldsymbol{f}(\boldsymbol{x}) \\ \boldsymbol{r}(\boldsymbol{x})\end{pmatrix}\right\}\end{aligned} \tag{6.2.4}$$

(see Sect. 3.2). The first criterion function is the *Integrated Mean Squared Prediction Error* (IMSPE) which is defined to be the MSPE "averaged" over $\mathcal{X}$ as follows:

$$\mathcal{I}\left(\mathcal{D},\widehat{y}\right) = \int_{\mathcal{X}} \frac{\text{MSPE}\left[\widehat{y}(\boldsymbol{x})\right]}{\sigma_Z^2}\, w(\boldsymbol{x})\, d\boldsymbol{x}, \tag{6.2.5}$$

where $w(\cdot)$ is a specified nonnegative weight function satisfying $\int_{\mathcal{X}} w(\boldsymbol{x})\, d\boldsymbol{x} = 1$. An n_s-point design $\mathcal{D}_{imspe}$ is *IMSPE-optimal* if it *minimizes* the IMSPE criterion function, i.e.,

$$\mathcal{I}\left(\mathcal{D}_{imspe},\widehat{y}\right) = \min_{\mathcal{D}\subset\mathcal{X}} \mathcal{I}\left(\mathcal{D},\widehat{y}\right).$$

Because the reciprocal of σ_Z^2 is a multiplicative factor in $\mathcal{I}(\mathcal{D}_{imspe},\widehat{y})$, the same $\mathcal{D}_{imspe}$ is optimal for all σ_Z^2. Using the formula for the MSPE given in (6.2.4), $\mathcal{I}(\mathcal{D},\widehat{y})$ can be written as

$$1 - \text{tr}\left(\begin{bmatrix}\boldsymbol{0} & \boldsymbol{F}^\top \\ \boldsymbol{F} & \boldsymbol{R}\end{bmatrix}^{-1}\int_{\mathcal{X}}\begin{bmatrix}\boldsymbol{f}(\boldsymbol{x})\boldsymbol{f}^\top(\boldsymbol{x}) & \boldsymbol{f}(\boldsymbol{x})\boldsymbol{r}^\top(\boldsymbol{x}) \\ \boldsymbol{r}(\boldsymbol{x})\boldsymbol{f}^\top(\boldsymbol{x}) & \boldsymbol{r}(\boldsymbol{x})\boldsymbol{r}^\top(\boldsymbol{x})\end{bmatrix} w(\boldsymbol{x})\, d\boldsymbol{x}\right). \tag{6.2.6}$$

The expression (6.2.6) simplifies under certain conditions. For example, suppose $\mathcal{X} = [a_1, b_1]\times\cdots\times[a_d, b_d]$; the i^{th} regressor $f_i(\boldsymbol{x})$ has the product form $\prod_j f_{ij}(x_j)$, $i = 1, \ldots, p$, for real-valued functions $f_{ij}(\cdot)$ or is the sum of such functions; the correlation $r_i(\boldsymbol{x})$ has the product form $\prod_j r_{ij}(x_j)$, $i = 1, \ldots, n_s$, for real-valued functions $r_{ij}(\cdot)$; and the weight function $w(\boldsymbol{x})$ has the product form $\prod_j w_j(x_j)$ for real-valued functions $w_j(\cdot)$. Then the multidimensional integral in (6.2.6) simplifies to a product of one-dimensional integrals, which is useful in reducing the amount of computation required by an optimal design algorithm. These conditions on $\boldsymbol{r}(\cdot)$ and $\boldsymbol{f}(\cdot)$ are satisfied if, for example, a polynomial regression model is used for the linear model portion of $Y(\cdot)$ and a product correlation structure is assumed, such as the product power exponential correlation (2.2.11).

Constructing an IMSPE-optimal design requires minimizing $\mathcal{I}(\mathcal{D},\widehat{y})$ as a function of the $n_s \times d$ inputs of the desired design. In practice, $n_s \times d$ can be large as researchers seek to have an adequate number of runs to explore simulators with control, environmental, and other types of inputs. Welch et al. (1992) used a quasi-Newton algorithm to search for an IMSPE-optimal design. Quasi-Newton algorithms are susceptible to being caught in a local optimum and must be used with multiple starting designs. Leatherman et al. (2014) found that using a direct search particle swarm algorithm was effective in identifying promising starting designs for the quasi-Newton algorithm. Nevertheless, the computation of IMSPE-optimal designs can only be accomplished for relatively small $n_s \times d$, say less than 150.

Figure 6.2 shows three 20-point IMSPE-optimal designs in $d = 2$ dimensions for a uniform weight function when $(x_1, x_2) \in [0, 1]^2$. Each design assumes the constant mean GP model with Gaussian correlation function

$$R(h_1, h_2) = \prod_{i=1}^{2} \exp(-\xi_i h_i^2),$$

where $\boldsymbol{\xi} = (\xi_1, \xi_2)^\top > 0$ is *known*. The three panels of Fig. 6.2 correspond to $\boldsymbol{\xi} = (1, 1)$ in panel (a), $\boldsymbol{\xi} = (25, 25)$ in panel (b), and $\boldsymbol{\xi} = (1, 25)$ in panel (c). Both (a) and (b) designs appear space-filling, but the points in (a) are closer to the boundary than those in (b). In case (c) the projections onto the x_2-axis are more "spread out" than the projections onto the x_1-axis. The behavior is similar to that seen in Fig. 6.1 for maximum entropy designs, and similar comments apply here. If the correlation is decidedly non-isotropic, the design may be sensitive to the actual values of the correlation parameters.

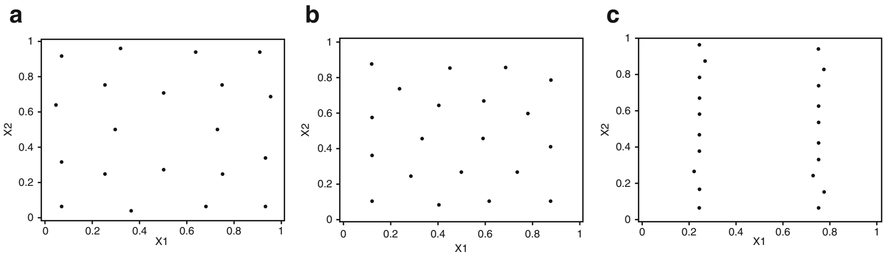


Fig. 6.2 IMSPE-optimal designs on $[0, 1]^2$ for the constant mean model obtained by minimizing (6.2.6) for the uniform weight function. In panel (**a**), $\boldsymbol{\xi} = (1, 1)$; in panel (**b**), $\boldsymbol{\xi} = (25, 25)$; and in panel (**c**), $\boldsymbol{\xi} = (1, 25)$

The IMSPE criterion function (6.2.5) requires knowledge of the correlation function $R(\cdot)$. In computer experiment applications, the correlation structure often depends on unknown parameters. Thus, in practice, the true IMSPE-optimal design cannot be computed. To overcome this problem, several approaches can be used to find approximate IMSPE designs. First, one can pick plausible values for the unknown parameters and pretend these are the true values. In the optimal design literature, such designs are referred to as locally optimal. Second, one can use a minimax approach. Find the design that minimizes the maximum MSPE, the maximum computed over all values of the unknown parameters. This approach will be discussed shortly. Third, one can employ a Bayesian approach, finding the design that minimizes the expected value of the MSPE, the expectation taken with respect to a prior on the unknown parameters. The Bayesian approach is discussed in Sect. 6.5.2. Fourth, one could use a two-stage procedure. In the first stage, data is collected using one of the designs discussed in Chap. 5, for example, an LHD. Based on the first-stage data, the unknown correlation parameters are estimated. The second stage determines an IMSPE-optimal design treating the estimates as the true parameter values.

The second MSPE-based criterion introduced by Sacks et al. (1989a) employed a *minimax approach* to find an approximate IMSPE design. Assume $\boldsymbol{x} \in [0, 1]^d$ and $y(\boldsymbol{x})$ can be described as a draw from a GP $Y(\boldsymbol{x})$ with *isotropic Gaussian correlation function*

$$R(h_1, \ldots, h_d) = \prod_{i=1}^{d} \exp(-\xi\, h_i^2)\,, \tag{6.2.7}$$

with *unknown* rate parameter $\xi > 0$. Sacks et al. (1989a) conducted a robustness study to identify a value of ξ for generating an IMSPE-optimal design that performs well over a range of possible ξ values. Let $\boldsymbol{a}_\xi(\cdot)$ denote the coefficient vector of the BLUP $\widehat{y}_\xi(\boldsymbol{x})$ in (3.2.7). This notation explicitly indicates the dependence of these coefficients on the correlation parameter.

To give additional details about this approach, suppose $\xi_2 \geq 0$ is assumed in computing the BLUP $\widehat{y}_{\xi_2}(\boldsymbol{x})$ but that $\xi_1 \geq 0$ is the true value of the correlation parameter governing $Y(\boldsymbol{x})$. Welch et al. (1992) showed that the MSPE of $\widehat{y}_{\xi_2}(\boldsymbol{x})$ can be calculated as

$$\text{MSPE}_{\xi_1}\left[\widehat{y}_{\xi_2}(\boldsymbol{x})\right] = \sigma_Z^2\left[1 + \boldsymbol{a}_{\xi_2}^\top(\boldsymbol{x})\boldsymbol{R}_{\xi_1}\boldsymbol{a}_{\xi_2}(\boldsymbol{x}) - 2\boldsymbol{a}_{\xi_2}^\top(\boldsymbol{x})\boldsymbol{r}_{\xi_1}(\boldsymbol{x})\right].$$

Consider the IMSPE criterion function

$$\mathcal{I}_{\xi_1}\left(\mathcal{D}_{\xi_2}, \widehat{y}_{\xi_2}\right) = \int_{\mathcal{X}} \frac{\text{MSPE}_{\xi_1}\left[\widehat{y}_{\xi_2}(\boldsymbol{x})\right]}{\sigma_Z^2}\, d\boldsymbol{x} \tag{6.2.8}$$

(assuming a uniform weight function $w(\cdot)$). The Sacks et al. (1989a) robust method of choosing ξ proceeds as follows:

1. Select Ξ to be a finite set of possible values for the correlation parameter. Both large and small parameter values should be included in Ξ.
2. Assume that the correlation parameter has value $\xi_A \in \Xi$. Set $\xi_1 = \xi_2 = \xi_A$ in (6.2.8) and calculate an IMSPE-optimal design $\mathcal{D}_{imspe}^{\xi_A}$. Do this for each $\xi_A \in \Xi$. The representation (6.2.6) for $\mathcal{I}_{\xi_A}(\mathcal{D}_{\xi_A}, \widehat{y}_{\xi_A})$ can be used to perform the required minimization, where $\boldsymbol{R} = \boldsymbol{R}_{\xi_A}$, $\boldsymbol{r}(\cdot) = \boldsymbol{r}_{\xi_A}(\cdot)$, and $w(\cdot) \equiv 1$.
3. For each $\xi_T \in \Xi$ and $\xi_A \in \Xi$ set $\xi_1 = \xi_T$ and $\xi_2 = \xi_A$ in (6.2.8) and calculate $\mathcal{I}_{\xi_T}(\mathcal{D}_{imspe}^{\xi_A}, \widehat{y}_{\xi_A})$. Here, ξ_T represents the *true* value of the correlation parameter governing $Y(\boldsymbol{x})$. Then $\mathcal{I}_{\xi_T}(\mathcal{D}_{imspe}^{\xi_T}, \widehat{y}_{\xi_T}) \leq \mathcal{I}_{\xi_T}(\mathcal{D}_{imspe}^{\xi_A}, \widehat{y}_{\xi_A})$. Define the *relative efficiency* of strategy $(\mathcal{D}_{imspe}^{\xi_A}, \widehat{y}_{\xi_A})$ under ξ_T as follows:

$$\text{eff}_{\xi_T}\left(\mathcal{D}_{imspe}^{\xi_A}, \widehat{y}_{\xi_A}\right) = \frac{\mathcal{I}_{\xi_T}\left(\mathcal{D}_{imspe}^{\xi_T}, \widehat{y}_{\xi_T}\right)}{\mathcal{I}_{\xi_T}\left(\mathcal{D}_{imspe}^{\xi_A}, \widehat{y}_{\xi_A}\right)}.$$

Calculate these efficiencies for each $\xi_T \in \Xi$ and $\xi_A \in \Xi$.

4. Choose the design $\mathcal{D}^{\xi_A^*}_{imspe}$ that is *robust* in the sense of maximizing the minimum efficiency:

$$\min_{\xi_T \in \Xi} \text{eff}_{\xi_T}\left(\mathcal{D}^{\xi_A^*}_{imspe}, \widehat{y}_{\xi_A^*}\right) = \max_{\xi_A \in \Xi} \min_{\xi_T \in \Xi} \text{eff}_{\xi_T}\left(\mathcal{D}^{\xi_A}_{imspe}, \widehat{y}_{\xi_A}\right).$$

Sacks et al. (1989a) considered the efficacy of their robustness approach in a series of examples. In the first of these, they find that the design $\mathcal{D}^1_{imspe}$ corresponding to $\xi = 1$ was most robust. This design reduced an empirical measure of integrated squared error by a factor of 8–10 relative to a competing strategy using least squares prediction based on a 3^2 factorial design.

The robustness approach to design construction assumes that predictions are to be made using test functions drawn from an isotropic GP, either the one used to form the predictor or one with different rate parameters. In a study of IMSPE-optimal designs with space-filling and other designs, Leatherman et al. (2018) compared designs based on their *empirical mean squared prediction error (EMSPE),*

$$\text{EMSPE}(\mathcal{D}) = \frac{1}{g}\sum_{i=1}^{g}\left(\widehat{y}^E(\boldsymbol{x}_i) - y(\boldsymbol{x}_i)\right)^2,$$

for a set of test bed functions, $y(\boldsymbol{x})$; the g prediction inputs, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_g$ are a set of "space-filling" inputs that are $y(\boldsymbol{x})$-specific. Some test bed functions are drawn from a (broader) class of GPs (than used by Sacks et al. (1989a)), while others are selections from various random parameter function classes including nonstationary ones. The Leatherman et al. (2018) comparisons concluded that using the IMSPE-optimal design corresponding to (6.2.7) with $\xi = -\ell n(0.5)$ $(\exp(-\xi) = 0.5)$ provides smaller prediction errors than do IMSPE-optimal designs corresponding to other correlation values and this IMSPE-optimal design is also superior to the widely used maximin LHD designs.

Picard and Williams (2013) utilized a multistage approach to IMSPE-optimal design in the context of an adaptive importance sampling algorithm for rare event estimation with simulators $y(\boldsymbol{x})$ allowing only a limited budget of runs. The goal is to estimate $P[y(\boldsymbol{X}) \geq T]$ for a given threshold T via importance sampling, with inputs $\boldsymbol{x}$ distributed according to a known distribution $[\boldsymbol{X}]$. A first-stage space-filling design $\mathcal{D}_1$ (such as an LHD) specifies simulator runs used to estimate the correlation parameters $\boldsymbol{\kappa}$ of the BLUP, resulting in the EBLUP $\widehat{y}(\boldsymbol{x} \mid \widehat{\boldsymbol{\kappa}}_1, \mathcal{D}_1)$. A parametric importance density $g_1(\cdot)$ (e.g., multivariate normal) is estimated from samples $\boldsymbol{X} \sim [\boldsymbol{X}]$ satisfying $\widehat{y}(\boldsymbol{X} \mid \widehat{\boldsymbol{\kappa}}_1, \mathcal{D}_1) \geq T$. A second-stage design $\mathcal{D}_2$ is then found that minimizes the criterion (6.2.5), calculated using $\widehat{\boldsymbol{\kappa}}_1$ and $\mathcal{D}_1 \cup \mathcal{D}_2$. The weight function $w(\boldsymbol{x})$ is taken to be the product of the marginal components $g_{1i}(x_i)$ of the importance density $g_1(\boldsymbol{x})$, allowing the reduction of (6.2.6) to a product of one-dimensional integrals. This design approach is motived by the desire to achieve targeted improvement of the BLUP's predictive capability in the region of input space responsible for rare events, thereby increasing the efficiency of importance density updates and ultimately estimation of $P[y(\boldsymbol{X}) \geq T]$. Once $\mathcal{D}_2$ is determined, new simulator runs are obtained, and the correlation parameters are updated to $\widehat{\boldsymbol{\kappa}}_2$. The EBLUP and importance density are updated to $\widehat{y}(\boldsymbol{x} \mid \widehat{\boldsymbol{\kappa}}_2, \mathcal{D}_1 \cup \mathcal{D}_2)$ and $g_2(\cdot)$, respectively. This process continues until minimal improvement in the estimate of $P[y(\boldsymbol{X}) \geq T]$ is achieved or the budget of simulator runs is exhausted.

There are other criteria for selecting designs that are based on the MSPE. This section concludes by describing one of these, the *maximum mean squared prediction error* criterion. As above, let $y(\boldsymbol{x})$ be an output function defined on input domain $\mathcal{X}$, $\mathcal{D}$ an n-point design, and $\widehat{y}(\boldsymbol{x})$ the BLUP of $y(\boldsymbol{x})$. The maximum mean squared prediction error (MMSPE) criterion function is defined as follows:

$$M\left(\mathcal{D}, \widehat{y}\right) = \max_{x \in \mathcal{X}} \frac{\text{MSPE}\left[\widehat{y}(\boldsymbol{x})\right]}{\sigma_Z^2} . \tag{6.2.9}$$

The design $\mathcal{D}_{mmpse}$ is *MMSPE-optimal* if it *minimizes* the worst case prediction error over $\mathcal{X}$, i.e., it minimizes

$$M\left(\mathcal{D}_{mmspe}, \widehat{y}\right) = \min_{\mathcal{D} \subset \mathcal{X}} M\left(\mathcal{D}, \widehat{y}\right) .$$

The MMSPE criterion function (6.2.9) depends on any unknown correlation parameters but is independent of the process variance. Hence the same design is optimal for all σ_Z^2. When $\mathcal{X}$ is a continuous region, the determination of an MMSPE-optimal design can be extremely computationally intensive, as a d-dimensional maximization of a complicated function is required for each design considered by an algorithm. As do IMSPE-optimal designs, MMSPE-optimal designs tend to locate points away from the boundary of the design space.

The IMSPE and MMSPE design criteria are generalizations of the classical A-optimality and G-optimality criteria. To see this, suppose that $Z(\boldsymbol{x})$ in (2.2.3) is a *white noise process*, so that $\boldsymbol{R} = \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix. Then $\boldsymbol{Y}^{n_s}$, the vector of the training data, follows the traditional regression model with random error used in response surface modeling for physical experiments. In this case, the IMSPE criterion function (6.2.5) simplifies to

$$\begin{aligned} \mathcal{I}\left(\mathcal{D}, \widehat{y}\right) - 1 &= \int_{\mathcal{X}} \boldsymbol{f}^\top(\boldsymbol{x})(\boldsymbol{F}^\top \boldsymbol{F})^{-1} \boldsymbol{f}(\boldsymbol{x})\, w(\boldsymbol{x})\, d\boldsymbol{x} \\ &= \text{tr}\left(\boldsymbol{W}(\boldsymbol{F}^\top \boldsymbol{F})^{-1}\right), \end{aligned}$$

where

$$\boldsymbol{W} = \int_{\mathcal{X}} \boldsymbol{f}(\boldsymbol{x}) \boldsymbol{f}^\top(\boldsymbol{x})\, w(\boldsymbol{x})\, d\boldsymbol{x}$$

is a positive definite $p \times p$ weight matrix. A design minimizing

$$\text{tr}\left(\boldsymbol{W}(\boldsymbol{F}^\top \boldsymbol{F})^{-1}\right)$$

is called *L-optimal* (see Pukelsheim (1993)). Thus, the IMSPE-optimal designs are the L-optimal designs when $Z(\boldsymbol{x})$ is a white noise process.

L-optimal designs are related to A-optimal designs as follows. Let $\boldsymbol{H}$ be a $p \times p$ square root of $\boldsymbol{W}$; that is, $\boldsymbol{W} = \boldsymbol{H}\boldsymbol{H}^\top$. Then,

$$\operatorname{tr}\left(\boldsymbol{W}(\boldsymbol{F}^\top\boldsymbol{F})^{-1}\right) = \operatorname{tr}\left(\boldsymbol{H}^\top(\boldsymbol{F}^\top\boldsymbol{F})^{-1}\boldsymbol{H}\right)$$

so a design is L-optimal if and only if it is *A-optimal* for the parameter subsystem $\boldsymbol{H}^\top\boldsymbol{\beta}$.

From Eq. (6.2.4), the MMSPE criterion function (6.2.9) simplifies to

$$M\left(\mathcal{D}, \widehat{y}\right) - 1 = \max_{\boldsymbol{x}\in\mathcal{X}} \boldsymbol{f}^\top(\boldsymbol{x})(\boldsymbol{F}^\top\boldsymbol{F})^{-1}\boldsymbol{f}(\boldsymbol{x})\,.$$

A design minimizing

$$\max_{\boldsymbol{x}\in\mathcal{X}} \boldsymbol{f}^\top(\boldsymbol{x})(\boldsymbol{F}^\top\boldsymbol{F})^{-1}\boldsymbol{f}(\boldsymbol{x})$$

is called *G-optimal*. Thus, the MMSPE-optimal designs are the G-optimal designs when $Z(\boldsymbol{x})$ is a white noise process.

In sum, our experience is that maximum entropy, IMSPE-optimal, and MMSPE-optimal designs are not used by practitioners as frequently as maximin LHDs, primarily because LHDs are relatively easy to construct for any sample size. However, IMSPE-optimal designs deserve strong consideration.

6.3 Designs Based on Optimization Criteria

6.3.1 Introduction

This subsection describes several sequential experimental design strategies that have been proposed for the important problem of finding an input $\boldsymbol{x} \in \mathcal{X}$ that "optimizes" the output of a computer simulator. Depending on the application, one or more simulators, $y_1(\boldsymbol{x}), \ldots, y_m(\boldsymbol{x})$ $(m \geq 1)$ may be of interest. In some applications, the $y_i(\cdot)$ are the objects of direct interest, while in other applications, either integrals or linear combinations of the $y_i(\cdot)$ are of interest. As illustrated in Sect. 1.2, an example of the latter occurs when $\boldsymbol{x}$ consists of both control and environmental components, i.e., $\boldsymbol{x} = (\boldsymbol{x}_c, \boldsymbol{x}_e)$. In this case, suppose that the environmental variables have a *known* probability distribution, which is specified either by the probability mass function $w_j = P\{\, \boldsymbol{X}_e = \boldsymbol{x}_{e,j}\,\}$ on n_e support points $\{\boldsymbol{x}_{e,j}\}$ or by the probability density function $w(\cdot)$. Then the quantities of interest are

$$\mu_i(\boldsymbol{x}_c) \equiv \sum_{j=1}^{n_e} w_j\, y_i(\boldsymbol{x}_c, \boldsymbol{x}_{e,j}) \quad \left(\text{or } \equiv \int y_i(\boldsymbol{x}_c, \boldsymbol{x}_e)\, w(\boldsymbol{x}_e)\, d\boldsymbol{x}_e\right), \tag{6.3.1}$$

which is the mean of $y_i(\cdot)$ with respect to the distribution of the $\boldsymbol{X}_e$ variables for $i = 1, \ldots, m$.

Sections 6.3.2–6.3.4 consider problems of optimizing a *single* output $y_1(\cdot)$ ($m = 1$); two specific problems of this sort are considered. The first is that of minimizing $y_1(\boldsymbol{x})$ as a function of *all* the input variables $\boldsymbol{x}$. In the second, $\boldsymbol{x}$ consists of control and environmental components, and the goal is to find control variable combinations $\boldsymbol{x}_{c,\min}$ that minimize $\mu_1(\cdot)$. Section 6.3.5 considers the case of multiple outputs ($m \geq 2$); it describes two methods of locating a minimizer $\boldsymbol{x}_{\min}$ of $y_1(\cdot)$ that satisfies feasibility constraints on $y_2(\cdot), \ldots, y_m(\cdot)$. It also presents a method for finding control variable combinations $\boldsymbol{x}_{c,\min}$ that minimize $\mu_1(\cdot)$, subject to feasibility constraints on $\mu_2(\cdot), \ldots, \mu_m(\cdot)$. Section 6.3.6 discusses Pareto optimization for multiple outputs.

The optimization algorithms presented in this section utilize multiple experimental design stages. The idea of these methods is to use first-stage data to obtain initial information about the *entire* response surface, while each additional stage takes account of all previous information to obtain an experimental design consistent with the optimization objective. A quantitative criterion is implemented by each algorithm for the purpose of deriving the experimental design in each stage.

6.3.2 Heuristic Global Approximation

Bernardo et al. (1992) proposed a sequential strategy for optimizing integrated circuit designs. Conceptually, their algorithm should be thought of as minimizing a single function, $y_1(\cdot)$, over inputs $\boldsymbol{x} \in \mathcal{X}$; thus this is an $m = 1$ problem. Their method sequentially refines the region of the input space where an optimum appears to be located. Computational considerations limit the number of model runs they could make and necessitate the use of a surrogate predictor for model output. In overview, the algorithm is implemented as follows.

1. Postulate an approximating model for the data-generating process.

Bernardo et al. (1992) adopted the stochastic process model of Sect. 2.3, with power exponential correlation function, which Sect. 3.3 has shown is substantially more flexible than standard polynomial models and implicitly accounts for nonlinearities in the inputs and complex interactions.

2. Design an initial experiment and collect the required data. Estimate model parameters and calculate the response predictor.

Bernardo et al. (1992) recommended initial designs containing *at least three runs* per *estimated model parameter* (in contrast, Loeppky et al. (2009) recommended larger initial designs having ten observations per *input variable*). Latin hypercube designs are run at the initial and each subsequent stage of this algorithm. As has been seen in Sect. 5.2.2, designs based on Latin hypercube samples have attractive marginal projection properties, while Sect. 5.4 shows that maximin distance LHDs provide a more uniform distribution of points for higher-dimensional projections than do randomly selected LHDs. Bernardo et al. (1992) used an empirical BLUP based on their stochastic process model as a computationally inexpensive surrogate for the true model output.

3. Check prediction accuracy and visualize the fitted models. If the prediction accuracy is sufficient, predict the global optimizer based on an EBLUP. Otherwise, go to Step 4.

Prediction accuracy is judged by computing the empirical root mean square prediction error (see (3.4.2)) for a set of randomly selected points in the analysis region and also by examining the range of predicted values. Thus this assessment of prediction accuracy depends on the particular application. In their circuit design example, Bernardo et al. (1992) compared the ERMSPE to the maximum allowable limit of variability in the reference current. In general, prediction accuracy is measured by the values of the ERMSPE relative to typical values of the response as dictated by the particular physical application under study.

4. Choose a subregion for the next experiment, and go to Step 1.

If additional stages are required, they are run on a *subregion* of the previous stage that contains the current *predicted* optimum. Sensitivity analysis methodology described in Sect. 7.1 provides a mechanism for quantifying the importance of input variables. In particular, main effect and interaction plots provide guidance for supplying reasonable input variable ranges to the next stage. Inactive inputs can be set equal to nominal values, which reduces the dimension of the input variable space in subsequent stages. *For model fitting, Bernardo et al. (1992) discarded data from previous stages for inputs falling outside the subregion of the current stage*. Once the response predictor attains the accuracy requirement, a confirmatory run is made at the location of the predicted optimum. If the actual results at the predicted optimum violate problem specifications, adjustments are made to the statistical model, and the algorithm continues until an acceptable solution is obtained. For example, in their integrated circuit study, Bernardo et al. (1992) added linear regression terms to the statistical model in response to a failed confirmatory run. This leads to increased prediction accuracy and a successful confirmatory run terminating the algorithm.

6.3.3 Mockus Criteria Optimization

Mockus et al. (1978) considered the problem of finding an optimal sequential algorithm to determine an $\boldsymbol{x}$ that minimizes the output, $y_1(\cdot)$, of a computer code. In terms of the introductory framework of Sect. 6.3.1, $m = 1$. The method assumes that $\boldsymbol{x}$ consists exclusively of control variables.

The algorithms studied by Mockus et al. (1978) assumed that the output is to be evaluated at a given number of input sites $N + 1$. Let $\mathcal{D}_n = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ and

$\boldsymbol{Y}_1^n = (y_1(\boldsymbol{x}_1), \ldots, y_1(\boldsymbol{x}_n))^\top$ denote the set of input sites at stage n (the experimental design) and the corresponding vector of outputs, respectively. The input selected at stage $(n+1)$ is allowed to depend on all the information that is previously available, i.e., $\boldsymbol{x}_{n+1} = d_n(\boldsymbol{Y}_1^n, \mathcal{D}_n)$ for $n = 1, \ldots, N$ and some function $d_n(\cdot, \cdot)$. Let $d_0(\cdot)$ be a function that specifies the initial input based on the information available at that stage, i.e., $\boldsymbol{x}_1 = d_0(\cdot)$. Formally, each $\boldsymbol{d} = (d_0, \ldots, d_N)$ is a sequential algorithm.

The statement of the criterion and the derivation of an optimal algorithm assumes that a prior $Y_1(\cdot)$ has been specified for $y_1(\cdot)$. Mockus et al. (1978) took this prior to be a stationary Gaussian process defined on the input space $\mathcal{X}$. An algorithm $\boldsymbol{d}^*$ is optimal, provided it minimizes the expected deviation of $Y_1(\boldsymbol{x}_{N+1})$ from the global minimum of $Y_1(\cdot)$ in the sense of achieving

$$\min_{\boldsymbol{d}} E\left[Y_1(\boldsymbol{x}_{N+1}(\boldsymbol{d})) - \min_{\boldsymbol{x} \in \mathcal{X}} Y_1(\boldsymbol{x})\right], \tag{6.3.2}$$

where $\boldsymbol{x}_{N+1}(\boldsymbol{d})$ denotes the $\boldsymbol{x}_{N+1}$ produced using algorithm $\boldsymbol{d}$. Note that the expectation in (6.3.2) is with respect to the $Y_1(\cdot)$ prior on $y_1(\cdot)$; also the argument of the expectation must be nonnegative, which makes the expectation itself nonnegative. Mockus et al. (1978) presented an N-dimensional dynamic program that, in principle, produces a $\boldsymbol{d}^*$ that attains (6.3.2). Unfortunately, this sequential algorithm is essentially computationally infeasible.

Mockus et al. (1978) and Mockus et al. (1994) proposed an alternative one-stage, myopic algorithm that mimics the operation of the full dynamic programming solution to (6.3.2). For a given input $\boldsymbol{x}_b$, design $\mathcal{D}_n$, and corresponding $\boldsymbol{Y}_1^n$, compute

$$\mu\left(\boldsymbol{x}_b \,\middle|\, \boldsymbol{Y}_1^n, Y_1(\boldsymbol{x}_a)\right) = E\left[Y_1(\boldsymbol{x}_b) \,\middle|\, \boldsymbol{Y}_1^n, Y_1(\boldsymbol{x}_a)\right],$$

where $Y_1(\boldsymbol{x}_a)$ is a hypothetical observation at the candidate input $\boldsymbol{x}_a$. This quantity is the mean output at $\boldsymbol{x}_b$ given current data *and* the hypothetical observation at $\boldsymbol{x}_a$. The Mockus update step selects $\boldsymbol{x}_{n+1}$ to satisfy

$$\boldsymbol{x}_{n+1} \in \arg\min_{\boldsymbol{x}_a \in \mathcal{X}} E\left[\min_{\boldsymbol{x}_b \in \mathcal{X}} \mu\left(\boldsymbol{x}_b \,\middle|\, \boldsymbol{Y}_1^n, Y_1(\boldsymbol{x}_a)\right) \,\middle|\, \boldsymbol{Y}_1^n\right].$$

In principle, the algorithm seeks an input $\boldsymbol{x}_a$ that achieves

$$\min_{\boldsymbol{x}_b \in \mathcal{X}} \mu\left(\boldsymbol{x}_b \,\middle|\, \boldsymbol{Y}_1^n, y_1(\boldsymbol{x}_a)\right).$$

However because $y_1(\boldsymbol{x}_a)$ is unknown, this output is approximated by replacing $y_1(\boldsymbol{x}_a)$ by $Y_1(\boldsymbol{x}_a)$ and computing the expectation of the resulting expression conditional on the available data, $\boldsymbol{Y}_1^n$. Iterations are executed until convergence is obtained. Unfortunately, this algorithm is also *computationally infeasible* for computationally expensive $y_1(\cdot)$ due to the need for repeated evaluation of $y_1(\cdot)$.

6.3.4 Expected Improvement Algorithms for Optimization

In contrast to the criteria described in the previous subsection (having difficult or impossible implementations), the next two subsections describe *implementable* but heuristic algorithms for three problems of global optimization. In this subsection, an algorithm is given for global minimization of an output $y_1(\boldsymbol{x})$ as a function of all the input variables $\boldsymbol{x}$. An extension of this algorithm to settings involving stochastic simulators $y_1(\cdot)$ is presented. Then a global optimization algorithm will be presented for the case when there are both control and environmental variables and the goal is to find the control variable combination that optimizes the mean output with respect to the environmental variables. In Sect. 6.3.5, constrained optimization algorithms will be discussed in the setting where there are *multiple* outputs and the goal is to minimize one of the outputs subject to constraints determined by the remaining outputs.

6.3.4.1 Schonlau and Jones Expected Improvement Algorithm

Schonlau et al. (1998) and Jones et al. (1998) developed sequential design strategies to find $\boldsymbol{x}_{\min} \in \arg\min_{\boldsymbol{x}\in\mathcal{X}}\, y_1(\boldsymbol{x})$, where $y_1(\boldsymbol{x})$ is an unknown function. Their basic sequential design, termed the *efficient global optimization* (EGO) algorithm, adds one input site at each stage and is initiated by computing $y_1(\boldsymbol{x})$ on a space-filling set of n inputs so that initial information about $y_1(\cdot)$ is available over a wide portion of the input space. An experimental design such as the maximin distance LHD (Sect. 5.4) is utilized for initial exploration of $y_1(\cdot)$. As usual, let $\boldsymbol{y}_1^n$ denote the vector of outputs corresponding to the initial n-point experimental design; these n runs are referred to as training data.

Schonlau et al. (1998) assumed a Gaussian prior for $y_1(\boldsymbol{x})$, say $Y_1(\boldsymbol{x})$, of the form on the right-hand side of (3.1.1) with process variance σ_1^2 and a uniform prior distribution for the regression parameters $\boldsymbol{\beta}$. Let $\boldsymbol{Y}_1^n$ denote the prior associated with the vector of outputs $\boldsymbol{y}_1^n$. Their algorithm is based on the fact that when σ_1^2 is known,

$$\left[Y_1\left(\boldsymbol{x}_0\right) \,\middle|\, \boldsymbol{Y}_1^n = \boldsymbol{y}_1^n\right] \sim N\left(\widehat{y}_1(\boldsymbol{x}_0), s_1^2(\boldsymbol{x}_0)\right), \tag{6.3.3}$$

where $\widehat{y}_1(\boldsymbol{x}_0)$ is the BLUP (3.2.7) and

$$s_1^2(\boldsymbol{x}_0) = \sigma_1^2\left\{1 - \boldsymbol{r}_0^\top \boldsymbol{R}^{-1}\boldsymbol{r}_0 + \boldsymbol{h}_0^\top\left(\boldsymbol{F}^\top\boldsymbol{R}^{-1}\boldsymbol{F}\right)^{-1}\boldsymbol{h}_0\right\} \tag{6.3.4}$$

is the mean squared prediction error of the BLUP. Here $\boldsymbol{h}_0 = \boldsymbol{f}_0 - \boldsymbol{F}^\top\boldsymbol{R}^{-1}\boldsymbol{r}_0$, while the vectors, $\boldsymbol{f}_0$ and $\boldsymbol{r}_0$, and the matrices, $\boldsymbol{F}$ and $\boldsymbol{R}$, are defined below (3.2.7). In practice, the correlation parameters and σ_1^2 are estimated by maximum likelihood or restricted maximum likelihood as, say, described in Sect. 3.3.2.

The algorithms in both papers are based on a measure of "improving" the minimum of $y_1(\boldsymbol{x})$ that is available after the output has been calculated for n inputs.

Let

$$y^n_{\min} = \min_{i=1,\ldots,n} y_1(\boldsymbol{x}_i)$$

denote the minimum output that has been evaluated after n runs of the computer code. Consider a *potential* site $\boldsymbol{x}$ at which to evaluate the code. Compared with the current smallest *known* minimum value of $y_1(\cdot)$, define the amount of improvement in $y_1(\cdot)$ at $\boldsymbol{x}$ to be *zero* if $y_1(\boldsymbol{x}) \geq y^n_{\min}$, i.e., $y_1(\boldsymbol{x})$ provides no improvement over $y^n_{\min}$. If $y_1(\boldsymbol{x}) < y^n_{\min}$, the amount of improvement at $\boldsymbol{x}$ is defined to be the difference $y^n_{\min} - y_1(\boldsymbol{x})$. Hence, in principle, the improvement at $\boldsymbol{x}$ is defined to be

$$\text{Improvement at } \boldsymbol{x} = \begin{cases} y^n_{\min} - y_1(\boldsymbol{x}), & y^n_{\min} - y_1(\boldsymbol{x}) > 0 \\ 0, & y^n_{\min} - y_1(\boldsymbol{x}) \leq 0\,. \end{cases} \tag{6.3.5}$$

As for the Mockus algorithm, this is an "in principle" definition because $y_1(\boldsymbol{x})$ is *unknown* (although $y^n_{\min}$ is known from the training data). However, there is information about the value of $y_1(\boldsymbol{x})$ that is available from the posterior of $Y_1(\boldsymbol{x})$ given $\boldsymbol{Y}^n_1$ in (6.3.3). Hence a probabilistically based *improvement function* can be defined by

$$I_n(\boldsymbol{x}) = \begin{cases} y^n_{\min} - Y_1(\boldsymbol{x}), & y^n_{\min} - Y_1(\boldsymbol{x}) > 0 \\ 0, & y^n_{\min} - Y_1(\boldsymbol{x}) \leq 0 \end{cases}, \tag{6.3.6}$$

for $\boldsymbol{x} \in \mathcal{X}$. The random variable $I_n(\boldsymbol{x})$ depends (solely) on the random quantity $Y_1(\boldsymbol{x})$. The algorithm below summarizes the amount of improvement possible at each input site $\boldsymbol{x}$ by its *expected improvement* with respect to the posterior distribution of $[Y_1(\boldsymbol{x}) \mid \boldsymbol{Y}^n_1]$.

Example 6.1. Suppose that the input space $\mathcal{X}$ is one-dimensional and that Fig. 6.3 shows the posterior densities of $[y^n_{\min} - Y_1(x) \mid \boldsymbol{Y}^n_1 = \boldsymbol{y}^n_1]$ for $x \in \{x_1, x_2, x_3\}$. Examining the conditional density of $y^n_{\min} - Y_1(x)$ shows that x_1 is a good candidate for exploration because this density is concentrated on positive values. This fact suggests that $y_1(x_1)$ is likely to be lower than $y^n_{\min}$. On the other hand, x_2 is not a promising site for exploration because the posterior density of $y^n_{\min} - Y_1(x_2)$ is concentrated on negative values. The posterior density of $y^n_{\min} - Y_1(x_3)$ has relatively heavy tails, and much of the support of the density is again over the positive numbers; thus, x_3 is also a reasonable candidate for exploration. The heavy tails of the posterior density of $y^n_{\min} - Y_1(x_3)$ are a result of the fact that the MSPE at site x_3, $s^2_1(x_3)$ from (6.3.4), is large. Taking another observation at x_3 would decrease the MSPE $s^2_1(x_3)$ based on all $n + 1$ outputs. In general, an input $\boldsymbol{x}$ for which $s^2_1(\boldsymbol{x})$ is large can yield large values of the *expected improvement.* ♦

It is straightforward to show that the *expected improvement* satisfies $E[I_n(\boldsymbol{x}) \mid \boldsymbol{Y}^n_1] = 0$ for $\boldsymbol{x}$ in the input training data $\mathcal{D}_n$. This result coincides with our intuition that there is no benefit in recomputing $y_1(\boldsymbol{x})$ at previously investigated $\boldsymbol{x}$. If $\boldsymbol{x} \notin \mathcal{D}_n$, some algebra shows that

$$E\left[I_n(\boldsymbol{x}) \,\middle|\, \boldsymbol{Y}^n_1\right] = s_1(\boldsymbol{x}) \left\{ \frac{y^n_{\min} - \widehat{y}_1(\boldsymbol{x})}{s_1(\boldsymbol{x})}\, \Phi\!\left(\frac{y^n_{\min} - \widehat{y}_1(\boldsymbol{x})}{s_1(\boldsymbol{x})}\right) + \phi\!\left(\frac{y^n_{\min} - \widehat{y}_1(\boldsymbol{x})}{s_1(\boldsymbol{x})}\right)\right\}$$

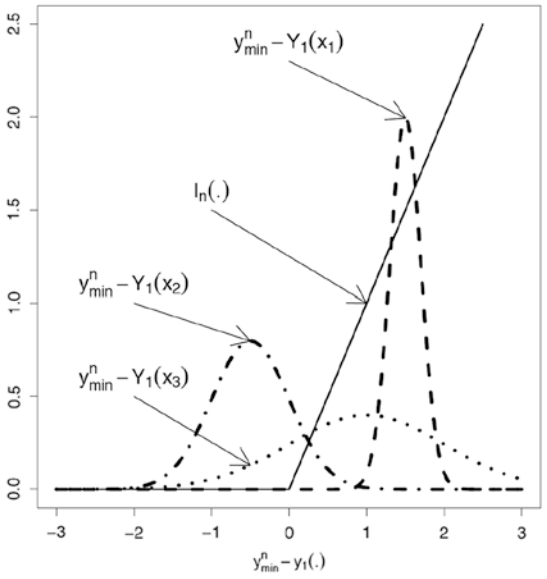


Fig. 6.3 The conditional (posterior) density of $y^n_{\text{min}} - Y_1(x)$ given $\boldsymbol{Y}^n_1$ for $x \in \{x_1, x_2, x_3\}$. The improvement function $I_n(\cdot)$ is plotted as a solid line

$$= (y^n_{\text{min}} - \widehat{y}_1(\boldsymbol{x}))\, \Phi\left(\frac{y^n_{\text{min}} - \widehat{y}_1(\boldsymbol{x})}{s_1(\boldsymbol{x})}\right) + s_1(\boldsymbol{x})\, \phi\left(\frac{y^n_{\text{min}} - \widehat{y}_1(\boldsymbol{x})}{s_1(\boldsymbol{x})}\right), \qquad (6.3.7)$$

where $\Phi(\cdot)$ and $\phi(\cdot)$ are the $N(0, 1)$ distribution and density function, respectively. By examining the terms in (6.3.7), it can be seen that the posterior expected improvement is "large" for those $\boldsymbol{x}$ having either

- a predicted value at $\boldsymbol{x}$ that is much smaller than the best minimum computed so far, i.e., $\widehat{y}_1(\boldsymbol{x}) \ll y^n_{\text{min}}$, *or*
- having much uncertainty about the value of $y_1(\boldsymbol{x})$, i.e., when $s_1(\boldsymbol{x})$ is large relative to $|y^n_{\text{min}} - \widehat{y}_1(\boldsymbol{x})|$.

These observations quantify the discussion in Example 6.1 about the operation of the algorithm. Candidate inputs are judged attractive if *either* there is high probability that their predicted output is below the current observed minimum *or* there is a large uncertainty (relative to $|y^n_{\text{min}} - \widehat{y}_1(\boldsymbol{x})|$) in the predicted output.

Starting with a space-filling design, the expected improvement algorithm updates the current input set $\mathcal{D}_n$ as follows.

Given the specified absolute tolerance ϵ_a, if

$$\max_{\boldsymbol{x} \in \mathcal{X}} E\left[I_n(\boldsymbol{x}) \mid \boldsymbol{Y}^n_1\right] < \epsilon_a,$$

then stop and predict $\boldsymbol{x}_{\text{min}}$ by an input site $\widehat{\boldsymbol{x}}_{\text{min}} \in \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ satisfying

$$y_1(\widehat{\boldsymbol{x}}_{\min}) = \min_{i=1,\ldots,n} y_1(\boldsymbol{x}_i)\,. \tag{6.3.8}$$

Otherwise, select an $\boldsymbol{x}_{n+1} \in \mathcal{X}$ to maximize $E[I_n(\boldsymbol{x}) \,|\, \boldsymbol{Y}_1^n]$. Set $\mathcal{D}_{n+1} = \mathcal{D}_n \cup \{\boldsymbol{x}_{n+1}\}$, $\boldsymbol{Y}_1^{n+1} = ((\boldsymbol{Y}_1^n)^\top, y_1(\boldsymbol{x}_{n+1}))^\top$, and increment n. Continue with the next update.

When the algorithm stops, instead of predicting $\boldsymbol{x}_{\min}$ by an input site $\widehat{\boldsymbol{x}}_{\min} \in \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ satisfying (6.3.8) one could predict $\boldsymbol{x}_{\min}$ by an input site $\widehat{\boldsymbol{x}}_{\min} \in \mathcal{X}$ that minimizes the $y_1(\boldsymbol{x})$ EBLUP based on $\mathcal{D}_n$.

Example 6.2 (EGO Algorithm Applied to the Branin Function). The Branin function has two inputs $\boldsymbol{x} = (x_1, x_2) \in \mathcal{X} = [-5, 10] \times [0, 15]$ and is given by the formula

$$y(x_1, x_2) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10.$$

Figure 6.4 plots the Branin function. As suggested by the figure, $y(x_1, x_2)$ has three global minima

$$y(\pi, 2.275) = y(3\pi, 2.475) = y(-\pi, 12.275) = 0.39789.$$

The EGO algorithm seeks to find *any* of these global minimizers.

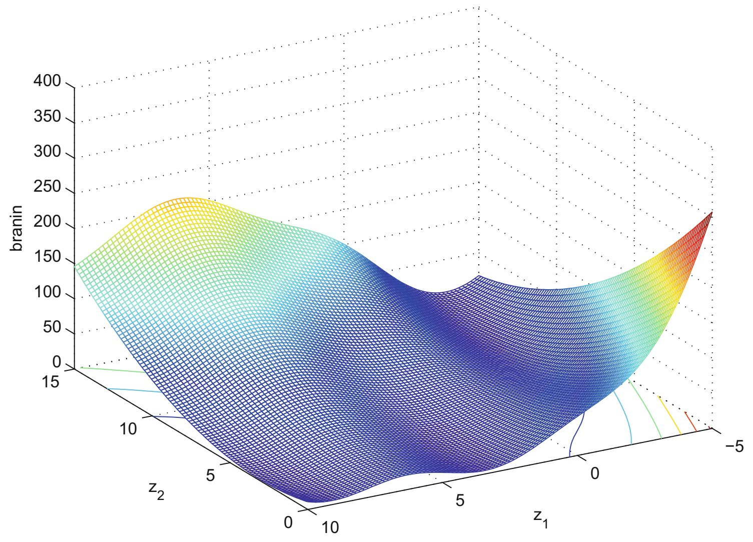


Fig. 6.4 The Branin function over the domain $[-5, 10] \times [0, 15]$

Starting with 21 input runs from a maximin LHD, the EGO algorithm was applied (as implemented in the SPACE package). SPACE added 12 points sequentially and then terminated. Figure 6.5 color codes these 33 (21 + 12) inputs, while Fig. 6.6 uses the same color coding to show the maximum expected improvement associated with the 12 added inputs. Figure 6.5 identifies four groups of inputs: the initial 21-point maximum LHD design (in blue); inputs #22 and #23 (in red) which explore parts of $\mathcal{X}$ where there is great uncertainty about the form of the function; inputs #24, #25, and #26 (in orange) which provide local searches near two of the global minima; and the final set of seven inputs, #27–#33 (in green) which explore the three global minima but provide little further decrease in the maximum expected improvement. The minimizing $\boldsymbol{x}$ selected by EGO is $\widehat{\boldsymbol{x}}_{\min} = (3.14042, 2.27273)$ with associated output $y(\widehat{\boldsymbol{x}}_{\min}) = 0.39790$ which has a relative error of $0.000025 = |0.39790 - 0.39789|/0.39789$. ♦

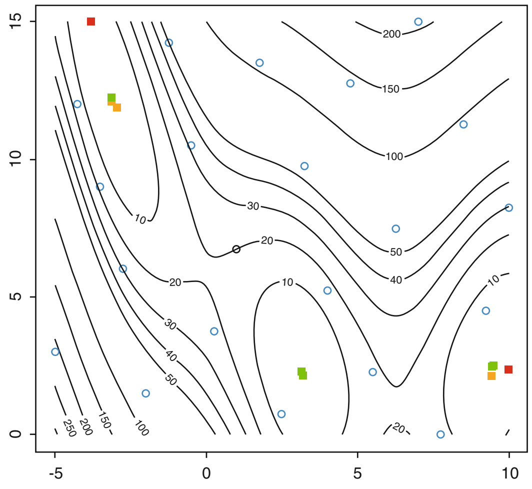


Fig. 6.5 The $[-5, 10] \times [0, 15]$ domain of the Branin function with a total of 33 inputs $\boldsymbol{x}$ at which $y(\boldsymbol{x})$ is calculated. Inputs #1–#21 (hollow blue circles) are the initial maximin LHD, and inputs #22–#33 are sequential EGO additions. The initial sequential inputs add #22 and #23 (solid red squares); inputs #24–#26 (solid orange squares) provide large decreases in maximum expected improvement; inputs #27–#33 (solid green squares) explore the three global minima but provide little further decrease in the maximum expected improvement. The maximum expected improvement for each of the 12 points added by EGO is shown in Fig. 6.6

Jones et al. (1998) (and Schonlau et al. (1998)) adopted the power exponential correlation structure of (2.2.11) for the Gaussian process $Y_1(\cdot)$ with one range and one smoothness parameter per input. They estimate these parameters by the method of maximum likelihood. Upon completion of each update step, the correlation parameters of the stochastic model can optionally be updated. The updating procedure can be computationally expensive, particularly for large designs.

Schonlau et al. (1998) provided a modification of the expected improvement algorithm that allows multiple input sites ("batches") to be generated at each stage (see also Schonlau (1997)). When multiple observations are taken in a stage, the

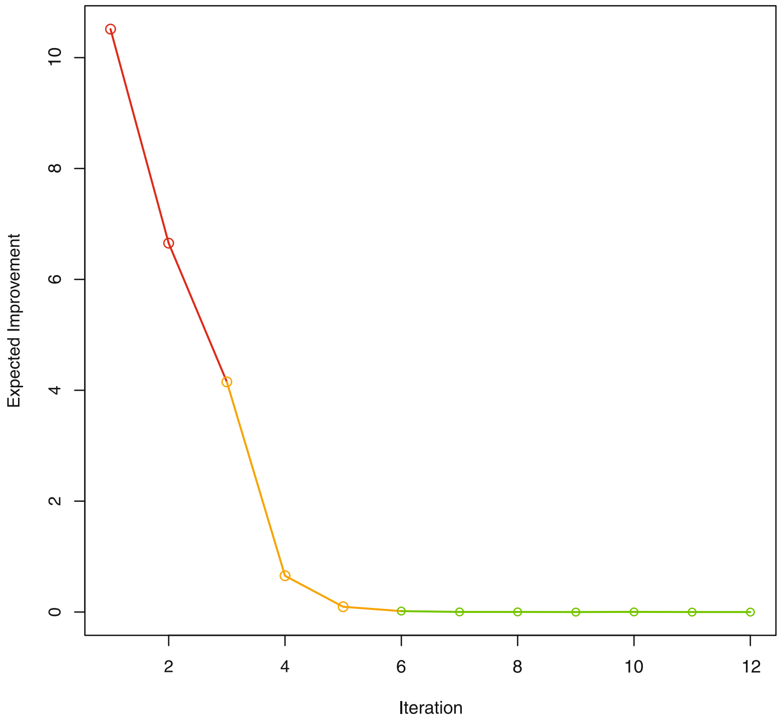


Fig. 6.6 The maximum expected improvement for each of the 12 points added by EGO which are color coded as in Fig. 6.5

correlation parameter updates take place at the end of the stage. Specifically, given a current design of size n and q iterates to be added, Schonlau et al. (1998) recommended updating the $s_1(\boldsymbol{x})$ coefficient in (6.3.7) after each iterate, but not updating the $s_1(\boldsymbol{x})$ term in the expressions $(y^n_{\min} - \widehat{y}_1(\boldsymbol{x}))/s_1(\boldsymbol{x})$. This heuristic forces all previously sampled inputs to be avoided, including the previous iterates of the current stage, as $s_1(\cdot)$ is 0 at these inputs. The empirical BLUP and MSPE are updated subsequent to the correlation parameters at the completion of each stage. If $\mathcal{X}$ is finite, the expected improvement algorithm will converge to the global minimum under the assumption that ϵ_a (or ϵ_r) = 0. Schonlau (1997) demonstrated the effectiveness of this algorithm for a suite of test problems where ϵ_a (or ϵ_r) > 0.

Schonlau et al. (1998) also considered a generalization of the expected improvement criterion in which the improvement (6.3.6) is replaced by

$$I^g_n(\boldsymbol{x}) = \begin{cases} (y^n_{\min} - Y_1(\boldsymbol{x}))^g, & \text{if } Y_1(\boldsymbol{x}) < y^n_{\min} \\ 0, & \text{otherwise} \end{cases},$$

for a given $g \in \{0, 1, 2, \ldots\}$. Larger values of g are associated with a more *global* search. This can be seen by examining Fig. 6.3. Provided $I_n(\boldsymbol{x}) \geq 1$, $I^{g_1}_n(\boldsymbol{x}) \geq I^{g_2}_n(\boldsymbol{x})$

for each input $\boldsymbol{x}$ when $g_1 \geq g_2$. Therefore, greater weight will be placed on the tails of the conditional distribution of $Y_1(\cdot)$ given $\boldsymbol{Y}_1^n$ for larger g values. Thus the global potential for large improvement is given increased quantitative importance. The quantity $E[I_n(\cdot) \,|\, \boldsymbol{Y}_1^n]$ in the stopping rule for the expected improvement algorithm is replaced by $E[I_n^g(\cdot) \,|\, \boldsymbol{Y}_1^n]^{1/g}$ so that the tolerance limits ϵ_a and ϵ_r have approximately the same interpretation for any g. Schonlau et al. (1998) provided recursive relations for computing $E[I_n^g(\boldsymbol{x}) \,|\, \boldsymbol{Y}_1^n]$.

6.3.4.2 Picheny Expected Quantile Improvement Algorithm

The EGO algorithm must be modified when the simulator $y_1(\boldsymbol{x})$ is stochastic to account for the outputs $y_1(\boldsymbol{x}_i)$ on the design $\mathcal{D}_n$ no longer being deterministic (i.e., repeated runs on the same design result in different outputs). The minimum output $y_{\min}^n$ on $\mathcal{D}_n$ is also no longer deterministic. To account for simulator stochasticity, the GP prior $Y_1(\cdot)$ for $y_1(\cdot)$ is augmented with an independent zero mean Gaussian white noise process (the *nugget effect*),

$$\widetilde{Y}_1(\boldsymbol{x}) = Y_1(\boldsymbol{x}) + \varepsilon_1(\boldsymbol{x}),$$

having covariance function

$$C_1(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sigma_Z^2 R_1(\boldsymbol{x}_1, \boldsymbol{x}_2) + \tau^2 I\{\boldsymbol{x}_1 = \boldsymbol{x}_2\}$$

with $R_1(\cdot,\cdot)$ a correlation function (see Sect. 2.2.2) and $I\{\cdot\}$ the indicator function that takes the value 1 if $\boldsymbol{x}_1 = \boldsymbol{x}_2$ and 0 otherwise. The nugget effect relaxes the exact interpolation property of GP models which is necessary for stochastic simulators. The value of τ^2 may vary by simulator run (e.g., the run at input $\boldsymbol{x}_i$ takes the value τ_i^2). In the discussion below, it is assumed that τ^2 is unknown but homogeneous throughout $\mathcal{X}$ and can therefore be estimated from the observed outputs $\widetilde{\boldsymbol{Y}}_1^n$ along with all other unknown parameters.

In this scenario, Picheny et al. (2013) proposed minimizing the *β-quantile* of the predictive distribution $[Y_1(\boldsymbol{x}) \,|\, \widetilde{\boldsymbol{Y}}_1^n]$, given by

$$q_n(\boldsymbol{x}) = \widehat{y}_1(\boldsymbol{x}) + \Phi^{-1}(\beta)\, s_1(\boldsymbol{x})$$

for $\Phi(\cdot)$ the standard normal distribution function and $\{\widehat{y}_1(\boldsymbol{x}), s_1(\boldsymbol{x})\}$ the mean and standard deviation of the predictive distribution $[Y_1(\boldsymbol{x}) \,|\, \widetilde{\boldsymbol{Y}}_1^n]$. The analog of $y_{\min}^n$ in this approach is $q_{\min}^n = \min\{q_n(\boldsymbol{x}_1), \ldots, q_n(\boldsymbol{x}_n)\}$, the minimum value of the β-quantile on $\mathcal{D}_n$. The quantity $q_{\min}^n$ is deterministic given observed data $\widetilde{\boldsymbol{Y}}_1^n$ on $\mathcal{D}_n$.

Letting $Q_{n+1}(\boldsymbol{x})$ denote the random β-quantile based on observed data $\widetilde{\boldsymbol{Y}}_1^n$ and unobserved future output $\widetilde{Y}_1(\boldsymbol{x})$, a probabilistically based *quantile improvement function* can be defined by

$$I_n^Q(\boldsymbol{x}) = \begin{cases} q_{\min}^n - Q_{n+1}(\boldsymbol{x}), & q_{\min}^n - Q_{n+1}(\boldsymbol{x}) > 0 \\ 0, & q_{\min}^n - Q_{n+1}(\boldsymbol{x}) \le 0 \end{cases}$$

for $\boldsymbol{x} \in \mathcal{X}$. The function $I_n^Q(\boldsymbol{x})$ represents the random amount the β-quantile would be reduced if the $(n+1)^{st}$ evaluation of the simulator is made at input $\boldsymbol{x}$. Because its value cannot be known precisely, the *expected quantile improvement (EQI) function* given previously calculated data $\widetilde{\boldsymbol{Y}}_1^n$ takes its place as the criterion for selecting the next run,

$$E\left[I_n^Q(\boldsymbol{x}) \;\middle|\; \widetilde{\boldsymbol{Y}}_1^n\right] = \left(q_{\min}^n - \widehat{y}_Q(\boldsymbol{x})\right) \Phi\left(\frac{q_{\min}^n - \widehat{y}_Q(\boldsymbol{x})}{s_Q(\boldsymbol{x})}\right) + s_Q(\boldsymbol{x})\, \phi\left(\frac{q_{\min}^n - \widehat{y}_Q(\boldsymbol{x})}{s_Q(\boldsymbol{x})}\right).$$

Here, $\phi(\cdot)$ is the standard normal density function, and $\{\widehat{y}_Q(\boldsymbol{x}),\, s_Q(\boldsymbol{x})\}$ are the mean and standard deviation of the predictive distribution $[Q_{n+1}(\boldsymbol{x}) \mid \widetilde{\boldsymbol{Y}}_1^n]$.

An input $\boldsymbol{x}_{n+1}$ satisfying

$$\boldsymbol{x}_{n+1} \in \arg\max_{\boldsymbol{x} \in \mathcal{X}} E\left[I_n^Q(\boldsymbol{x}) \;\middle|\; \widetilde{\boldsymbol{Y}}_1^n\right]$$

is selected as the next design point at which to run the simulator. This process continues until the expected quantile improvement is sufficiently small or a budget of N allowable simulator runs is expended.

As with EGO, the first term of $E[I_n^Q(\boldsymbol{x}) \mid \widetilde{\boldsymbol{Y}}_1^n]$ targets local search ($\widehat{y}_Q(\cdot) \ll q_{\min}^n$ favored), while the second term targets global search ($s_Q(\cdot)$ large). A quantile $\beta \ge 0.5$ is chosen to evaluate the EQI criterion. Calculation of EQI requires specification of the nugget effect associated with the unobserved future output $\widetilde{Y}_1(\boldsymbol{x})$. Picheny et al. (2013) suggested setting this *future variance* to $\tau^2/(N-n)$ when there are n runs in the current design. EQI will initially explore the space of designs globally and transition to a more local search as the simulation budget is consumed. If simulator runs are deterministic leading to adoption of a GP prior having no nugget effect, the EQI algorithm simplifies to EGO.

6.3.4.3 Williams Environmental Variable Mean Optimization

Williams et al. (2000) extended the expected improvement algorithm to input settings involving both control inputs, $\boldsymbol{x}_c$, and environmental inputs, $\boldsymbol{x}_e$. Their goal was to find a control variable input that minimizes the mean, $\mu_1(\cdot)$, of $y_1(\boldsymbol{x}_c, \boldsymbol{x}_e)$ averaged over $\boldsymbol{x}_e$ support points (the left-hand side of (6.3.1)). The proposed sequential design is especially useful in applications with "expensive" $y_1(\cdot)$ outputs.

Assume, for simplicity, that the input domain of $\boldsymbol{x} = (\boldsymbol{x}_c, \boldsymbol{x}_e)$ has the form $\mathcal{X} = \mathcal{X}_c \times \mathcal{X}_e$ where $\mathcal{X}_c$ denotes the domain of the control variables and $\mathcal{X}_e$ denotes the domain of the environmental variables. Assume that the environmental variable is discrete with a finite number, n_e, of support points. Let $\boldsymbol{x}_{e,j}$, $j = 1, \ldots, n_e$, denote the support points and $\{w_j\}$, $j = 1, \ldots, n_e$ the corresponding probabilities (weights).

The mean $\mu_1(\boldsymbol{x}_c) = \sum_{j=1}^{n_e} w_j\, y_1(\boldsymbol{x}_c, \boldsymbol{x}_{e,j})$ is the weighted output $y_1(\boldsymbol{x}_c, \boldsymbol{x}_{e,j})$ summed over the n_e support values; $\mu_1(\boldsymbol{x}_c)$ inherits the prior process defined by

$$M_1(\boldsymbol{x}_c) \equiv \sum_{j=1}^{n_e} w_j\, Y_1(\boldsymbol{x}_c, \boldsymbol{x}_{e,j}),$$

where $Y_1(\boldsymbol{x}_c, \boldsymbol{x}_e)$ has the Gaussian prior used by Schonlau et al. (1998) and Jones et al. (1998).

Let $\mathcal{D}_n = \{(\boldsymbol{x}_{c,i}, \boldsymbol{x}_{e,i}),\ 1 \le i \le n\}$ denote a generic n-point experimental design; $\{\boldsymbol{x}_{c,i},\ 1 \le i \le n\}$ denotes the control variable portions of this design. Williams et al. (2000) based their algorithm on the theoretical analog of (6.3.5)

$$\text{Improvement at } \boldsymbol{x}_c = \begin{cases} \mu^n_{\min} - \mu_1(\boldsymbol{x}_c), & \mu^n_{\min} - \mu_1(\boldsymbol{x}_c) > 0 \\ 0, & \mu^n_{\min} - \mu_1(\boldsymbol{x}_c) \le 0 \end{cases},$$

where $\mu^n_{\min} = \min_{i=1,\ldots,n} \mu_1(\boldsymbol{x}_{c,i})$. Notice that, in contrast to the known value of $y^n_{\min}$ used in (6.3.6), $\mu^n_{\min}$ is never directly calculated (as is the case with $\mu_1(\cdot)$). The corresponding probability-based *improvement function* is

$$I_n(\boldsymbol{x}_c) = \begin{cases} M^n_{\min} - M_1(\boldsymbol{x}_c), & M^n_{\min} - M_1(\boldsymbol{x}_c) > 0 \\ 0, & M^n_{\min} - M_1(\boldsymbol{x}_c) \le 0 \end{cases},$$

where $M^n_{\min} = \min_{i=1,\ldots,n} M_1(\boldsymbol{x}_{c,i})$. The prior (and posterior) of $M^n_{\min}$ is used to predict $\mu^n_{\min}$. Thus the estimation of the expected improvement is more challenging than in Schonlau et al. (1998) or Jones et al. (1998).

In outline, the Williams et al. (2000) algorithm to minimize $\mu_1(\cdot)$ consists of the following steps:

1. *Select an initial experimental design $\mathcal{D}_n$, and compute the vector of model outputs, $\boldsymbol{Y}^n_1$, at each design point.*
2. *Estimate the vector of correlation parameters (by, say, (restricted) maximum likelihood or under the Bayesian framework by the mode of their joint posterior distribution).*
3. *Select $\boldsymbol{x}_{c,n+1} \in \mathcal{X}_c$ to maximize the expected improvement,*

$$\boldsymbol{x}_{c,n+1} \in \arg\max_{\boldsymbol{x}_c \in \mathcal{X}_c} E\left[I_n(\boldsymbol{x}_c) \,\middle|\, \boldsymbol{Y}^n_1 \right].$$

4. *Given $\boldsymbol{x}_{c,n+1}$, select $\boldsymbol{x}_{e,n+1} \in \mathcal{X}_e$ to minimize the mean squared prediction error given $\boldsymbol{Y}^n_1$,*

$$\boldsymbol{x}_{e,n+1} \in \arg\min_{\boldsymbol{x}_e \in \mathcal{X}_e} E\left[\left(\widehat{M}^{n+1}_1(\boldsymbol{x}_{c,n+1}) - M_1(\boldsymbol{x}_{c,n+1}) \right)^2 \,\middle|\, \boldsymbol{Y}^n_1 \right],$$

where $\widehat{M}_1^{n+1}(\cdot)$ is the conditional mean of $M_1(\cdot)$ based on the data $\boldsymbol{Y}_1^n$ and the latent observation $Y_1(\boldsymbol{x}_{c,n+1}, \boldsymbol{x}_e)$.

5. *Determine if the algorithm should be stopped. If so the global minimizer in the control variable space is predicted to be the global minimizer of the conditional mean of $M_1(\cdot)$ based on the data $\boldsymbol{Y}_1^n$. If not, set $\mathcal{D}_{n+1} = \mathcal{D}_n \cup \{(\boldsymbol{x}_{c,n+1}, \boldsymbol{x}_{e,n+1})\}$, calculate the code $y_1(\cdot)$ at $(\boldsymbol{x}_{c,n+1}, \boldsymbol{x}_{e,n+1})$, increment n to $n + 1$, and go to Step 2 (correlation parameter estimation).*

The correlation parameter estimation in Step 2 can be extremely time-consuming, particularly for "large" experimental designs and/or high-dimensional inputs. A sensible modification of this algorithm is to update the correlation parameters frequently in the initial iterations and reduce the update rate as additional design points are added. In this way, the correlation parameters are repeatedly updated at the stage of the algorithm when they are least stable and the most substantial learning about the response surface occurs. As the response surface in the region of the optimum becomes predicted more accurately, correlation parameter updates become less necessary.

The expected improvement required in Step 3 cannot be expressed in closed form because $M_{\min}^n$ is not known. However, because the posterior of $M_{\min}^n$ given $\boldsymbol{Y}_1^n$ is known, Williams et al. (2000) presented a Monte Carlo algorithm for approximating $E[I_n(\boldsymbol{x}_c) \mid \boldsymbol{Y}_1^n]$. The mean squared error of prediction criterion for environmental variable selection, required in Step 4 of the algorithm, has a computationally convenient closed form.

Because the correlation parameters are re-estimated at each stage, the sequence of maximum expected improvements need not be monotone decreasing. Thus the stopping rules recommended in Williams et al. (2000) are based on observing a sequence of "small" maximum expected improvements relative to the history of such improvements established as the algorithm progresses. For example, moving averages and ranges of groups of observed expected improvements can be tracked, and the algorithm stopped, when these statistics reach a problem-specific threshold established relative to their initial observed values. The stopping criteria should ideally be met by two or more successive values of the moving average and range, suggesting stabilization of the expected improvement sequence.

6.3.5 Constrained Global Optimization

Schonlau et al. (1998) proposed an algorithm based on expected improvement to solve problems of constrained optimization. In this section, it is convenient to label the output functions $y_1(\cdot), \ldots, y_{k+1}(\cdot)$ so that $m = k + 1$ in our general notation. With this notation, the goal is to minimize $y_1(\boldsymbol{x})$ subject to $\boldsymbol{x}$ satisfying k constraints $l_i \leq y_i(\boldsymbol{x}) \leq 0$ for $i = 2, \ldots, k + 1$. The algorithm below requires all outputs be computed at each input. Let $(Y_1(\boldsymbol{x}), \ldots, Y_{k+1}(\boldsymbol{x}))$ denote the stochastic process model for $(y_1(\boldsymbol{x}), \ldots, y_{k+1}(\boldsymbol{x}))$.

Schonlau et al. (1998) proposed using the (probability-based) improvement function

$$I^g_{c,n}(\boldsymbol{x}) = \begin{cases} (y^n_{\min} - Y_1(\boldsymbol{x}))^g, & Y_1(\boldsymbol{x}) < y^n_{\min} \text{ and } l_i \leq Y_i(\boldsymbol{x}) \leq u_i \\ & \text{for } i = 2, \ldots, k+1 \\ 0, & \text{otherwise} \end{cases} \tag{6.3.9}$$

so that any constraint violation leads to zero improvement. As usual, $y^n_{\min}$ is the minimum of the $y_1(\boldsymbol{x})$ responses observed on the current experimental design. Schonlau et al. (1998) assumed that the objective and constraint processes are *mutually independent*; under this assumption, the conditional expected improvement is given by

$$E\left[I^g_{c,n}(\boldsymbol{x}) \,\middle|\, \boldsymbol{Y}^n_1, \ldots, \boldsymbol{Y}^n_{k+1}\right] = E\left[I^g_n(\boldsymbol{x}) \,\middle|\, \boldsymbol{Y}^n_1\right] \times$$
$$\mathrm{P}\left[l_2 \leq Y_2(\boldsymbol{x}) \leq u_2 \,\middle|\, \boldsymbol{Y}^n_2\right] \times \cdots \times \mathrm{P}\left[l_{k+1} \leq Y_{k+1}(\boldsymbol{x}) \leq u_{k+1} \,\middle|\, \boldsymbol{Y}^n_{k+1}\right],$$

where conditionally given the observed data $\boldsymbol{Y}^n_i$, $Y_i(\boldsymbol{x})$ has a Gaussian distribution which is the analog of (6.3.3).

Gramacy et al. (2016) proposed an expected improvement algorithm for constrained optimization of complex simulators motivated by the augmented Lagrangian numerical optimization framework (see Nocedal and Wright (2006)). The goal is to minimize $y_1(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathcal{X}$ subject to $\boldsymbol{x}$ satisfying k constraints $y_i(\boldsymbol{x}) \leq 0$ for $i = 2, \ldots, k+1$. This is accomplished by minimizing the *augmented Lagrangian*,

$$L_A(\boldsymbol{x}; \boldsymbol{\lambda}, \rho) = y_1(\boldsymbol{x}) + \sum_{i=1}^{k} \lambda_i\, y_{i+1}(\boldsymbol{x}) + \frac{1}{2\rho} \sum_{i=1}^{k} \max\{0, y_{i+1}(\boldsymbol{x})\}^2 \,,$$

where $\rho > 0$ is a *penalty parameter* and $\lambda_i \geq 0$ for $i = 1, \ldots, k$ are *Lagrange multipliers*.

The optimization algorithm is iterative: Given $(\rho^{j-1}, \boldsymbol{\lambda}^{j-1})$, the subproblem

$$\min_{\boldsymbol{x}} \left\{ L_A(\boldsymbol{x}; \boldsymbol{\lambda}^{j-1}, \rho^{j-1}) \;:\; \boldsymbol{x} \in \mathcal{X} \right\} \tag{6.3.10}$$

is solved, yielding candidate solution $\boldsymbol{x}_j$. The penalty parameter and Lagrange multipliers are then updated as follows,

$$\lambda^j_i = \max\left\{0, \lambda^{j-1}_i + \frac{1}{\rho^{j-1}}\, y_{i+1}(\boldsymbol{x}_j)\right\}, \quad i = 1, \ldots, k$$

$$\rho^j = \begin{cases} \rho^{j-1}, & \text{if } y_{i+1}(\boldsymbol{x}_j) \leq 0 \text{ for all } i = 1, \ldots, k \\ \frac{1}{2}\rho^{j-1}, & \text{otherwise} \end{cases},$$

and the iterations continue until user-monitored stopping criteria are met.

This optimization algorithm is prohibitively expensive to implement using only direct calculations of the objective and constraint functions due to the computational expense required. Application of the EGO algorithm to the augmented Lagrangian $L_A(\cdot)$ thus presents itself as an option for achieving computational feasibility. The process representation of $L_A(\cdot)$ takes the following form. Given $(\boldsymbol{\lambda}, \rho)$,

$$Y(\boldsymbol{x}) = Y_1(\boldsymbol{x}) + \sum_{i=1}^{k} \lambda_i \, Y_{i+1}(\boldsymbol{x}) + \frac{1}{2\rho} \sum_{i=1}^{k} \max\{0, Y_{i+1}(\boldsymbol{x})\}^2 \, . \tag{6.3.11}$$

The EGO algorithm could then be implemented with the expected improvement (6.3.7), using $Y(\cdot)$ from (6.3.11) rather than $Y_1(\cdot)$. However, Gramacy et al. (2016) discuss limitations associated with using the EGO algorithm directly on $L_A(\cdot)$. In particular, nonstationary GP models are likely needed to provide an adequate surrogate for $L_A(\cdot)$ due to the square and max operations involved in its computation.

Instead, Gramacy et al. (2016) recommend building individual GP surrogates for $Y_1(\cdot), Y_2(\cdot), \ldots, Y_{k+1}(\cdot)$, assuming these processes are mutually independent, and calculating a Monte Carlo estimate of expected improvement based on (6.3.6). Suppose calculations of the objective and constraint functions have already been made for n input vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, generating data vectors $\boldsymbol{Y}_1^n = \boldsymbol{y}_1^n, \ldots, \boldsymbol{Y}_{k+1}^n = \boldsymbol{y}_{k+1}^n$ along with the induced evaluations of $L_A(\cdot)$ at these inputs, denoted $\boldsymbol{Y}^n = \boldsymbol{y}^n$. Consider N samples generated from the joint distribution

$$\begin{aligned}\left[Y_1(\boldsymbol{x}), Y_2(\boldsymbol{x}), \ldots, Y_{k+1}(\boldsymbol{x}) \,\middle|\, \boldsymbol{y}_1^n, \boldsymbol{y}_2^n, \ldots, \boldsymbol{y}_{k+1}^n\right] &= \\ \left[Y_1(\boldsymbol{x}) \,\middle|\, \boldsymbol{Y}_1^n = \boldsymbol{y}_1^n\right] &\times \cdots \times \left[Y_{k+1}(\boldsymbol{x}) \,\middle|\, \boldsymbol{Y}_{k+1}^n = \boldsymbol{y}_{k+1}^n\right],\end{aligned}$$

where equality holds due to mutual independence of the individual processes. Denoting these samples by $Y_i^j(\boldsymbol{x})$ for $i = 1, \ldots k+1$, $j = 1, \ldots, N$, and letting $y_{\min}^n$ be the smallest observed value of the augmented Lagrangian (i.e., the smallest element of $\boldsymbol{y}^n$), the expected improvement is estimated as

$$\widehat{E}\left[I_n(\boldsymbol{x}) \,\middle|\, \boldsymbol{y}_1^n, \boldsymbol{y}_2^n, \ldots, \boldsymbol{y}_{k+1}^n\right] = \frac{1}{N} \sum_{j=1}^{N} \left(y_{\min}^n - Y^j(\boldsymbol{x})\right) I\left\{Y^j(\boldsymbol{x}) < y_{\min}^n\right\}$$

where $I\{\cdot\}$ is the indicator function and

$$Y^j(\boldsymbol{x}) = Y_1^j(\boldsymbol{x}) + \sum_{i=1}^{k} \lambda_i \, Y_{i+1}^j(\boldsymbol{x}) + \frac{1}{2\rho} \sum_{i=1}^{k} \max\left\{0, Y_{i+1}^j(\boldsymbol{x})\right\}^2 \, .$$

Maximization of $\widehat{E}[I_n(\boldsymbol{x}) \,|\, \boldsymbol{y}_1^n, \boldsymbol{y}_2^n, \ldots, \boldsymbol{y}_{k+1}^n]$ with respect to $\boldsymbol{x}$ replaces the subproblem (6.3.10) in the iterations of the above optimization algorithm.

Both of these approaches to constrained optimization can be extended to accommodate correlated constraint functions. The expanded statistical model requires a valid cross-correlation structure as discussed in Sect. 2.5. The additional model complexity increases the computational burden involved in fitting the statistical model, with possible benefits including increased flexibility in modeling the physical system and increased efficiency in finding a constrained optimum.

One paper that models the objective and constraint functions as dependent is Williams et al. (2010). Their methodology assumes that there is a *single* constraint function. Let $y_1(\boldsymbol{x}_c, \boldsymbol{x}_e)$ and $y_2(\boldsymbol{x}_c, \boldsymbol{x}_e)$ denote the simulator objective and simulator constraint functions, respectively. As earlier in the section, the environmental input is assumed to have n_e support points; the j^{th} input denoted $\boldsymbol{x}_{e,j}$ has associated probability w_j, $j = 1, \ldots n_e$. Let $\mu_1(\boldsymbol{x}_c)$ and $\mu_2(\boldsymbol{x}_c)$ denote the means of $y_1(\boldsymbol{x}_c, \boldsymbol{x}_e)$ and $y_2(\boldsymbol{x}_c, \boldsymbol{x}_e)$ with respect to the environmental input distribution. The goal is

$$\begin{gathered}\text{minimize} \quad \mu_1(\boldsymbol{x}_c) \\ \text{subject to} \\ \mu_2(\boldsymbol{x}_c) \le B\,.\end{gathered}$$

Williams et al. (2010) modeled $(y_1(\boldsymbol{x}), y_2(\boldsymbol{x}))$ as a draw from a *bivariate spatial autoregressive process* $(Y_1(\boldsymbol{x}), Y_2(\boldsymbol{x}))$ (see (2.5.7)). The $\boldsymbol{Y}$ prior induces the distribution

$$M_i(\boldsymbol{x}_c) \equiv \sum_{j=1}^{n_e} w_j\, Y_i(\boldsymbol{x}_c, \boldsymbol{x}_{e,j})$$

on $\mu_i(\boldsymbol{x}_c)$, $i = 1, 2$. Their algorithm is based on the modification

$$I_{c,n}(\boldsymbol{x}) = \begin{cases} M_{\min}^{n\star} - M_1(\boldsymbol{x}_c), & M_1(\boldsymbol{x}_c) < M_{\min}^{n\star} \text{ and } M_2^{0.05}(\boldsymbol{x}_c) \le B \\ 0, & \text{otherwise} \end{cases}$$

of the improvement (6.3.9) where $M_2^{0.05}(\boldsymbol{x}_c)$ denotes the 0.05 quantile of the distribution of $M_2(\boldsymbol{x}_c)$ and $M_{\min}^{n\star} = \min\{\, M_1(\boldsymbol{x}_{c,i}) : 1 \le i \le n \text{ and } M_2^{0.05}(\boldsymbol{x}_{c,i}) \le B\,\}$. The idea is that values of $M_1(\boldsymbol{x}_{c,i})$ from the current experimental design are included in the computation of the M_1 minimum only if there is strong evidence that $\boldsymbol{x}_{c,i}$ satisfies the constraint. This restriction is meant to enhance the chance of finding a global minimum of $\mu_1(\cdot)$ at which the constraint on $\mu_2(\cdot)$ holds.

The algorithm proceeds in a fashion similar to the single-objective optimization algorithm of Sect. 6.3.4. Once the algorithm is stopped, the constrained optimizer is predicted by solving the constrained optimization problem with the EBLUPs of $M_1(\cdot)$ and $M_2(\cdot)$. The correlation parameters can be intermittently estimated as the algorithm progresses, substantially reducing computation time while sacrificing little in terms of prediction accuracy. The Williams et al. (2010) algorithm extends without difficulty to lower bound or two-sided constraints.

6.3.6 Pareto Optimization

Assume that m simulator outputs $\boldsymbol{y}(\boldsymbol{x}) = (y_1(\boldsymbol{x}), \ldots, y_m(\boldsymbol{x}))^\top$, $m \geq 2$, can be calculated for $\boldsymbol{x}$ in a common input space $\mathcal{X}$. Let $\mathcal{Y}$ denote the range of $\boldsymbol{y}(\boldsymbol{x})$ as $\boldsymbol{x}$ varies over $\mathcal{X}$. It is desired to find a (single) $\boldsymbol{x}$ that simultaneously minimizes all $y_i(\boldsymbol{x})$, $i = 1, \ldots, m$. Typically, this objective is not possible. Instead, the following framework named after Vilfredo Pareto is adopted. Pareto used it to identify *compromise* choices of $\boldsymbol{x}$ for simultaneous "minimization" of a set of outputs.

Define a partial ordering of $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$ by saying that $\boldsymbol{x}_1$ *weakly dominates* $\boldsymbol{x}_2$, denoted $\boldsymbol{x}_1 \succeq \boldsymbol{x}_2$, if $y_i(\boldsymbol{x}_1) \leq y_i(\boldsymbol{x}_2)$ for all $i = 1, \ldots, m$. If at least one inequality $y_i(\boldsymbol{x}_1) < y_i(\boldsymbol{x}_2)$ is strict, then $\boldsymbol{x}_1$ is said to *dominate* $\boldsymbol{x}_2$, denoted $\boldsymbol{x}_1 \succ \boldsymbol{x}_2$. Similarly, for $\boldsymbol{y}(\boldsymbol{x}_1)$ and $\boldsymbol{y}(\boldsymbol{x}_2)$ in $\mathcal{Y}$, it will be said that $\boldsymbol{y}(\boldsymbol{x}_1)$ *weakly dominates* $\boldsymbol{y}(\boldsymbol{x}_2)$, $\boldsymbol{y}(\boldsymbol{x}_1) \succeq \boldsymbol{y}(\boldsymbol{x}_2)$, if $y_i(\boldsymbol{x}_1) \leq y_i(\boldsymbol{x}_2)$ for all $i = 1, \ldots, m$. If at least one inequality is strict, then $\boldsymbol{y}(\boldsymbol{x}_1)$ is said to *dominate* $\boldsymbol{y}(\boldsymbol{x}_2)$, denoted $\boldsymbol{y}(\boldsymbol{x}_1) \succ \boldsymbol{y}(\boldsymbol{x}_2)$.

An input $\boldsymbol{x} \in \mathcal{X}$ is *Pareto optimal* if and only if there is no $\boldsymbol{x}' \in \mathcal{X}$ such that $\boldsymbol{x} \prec \boldsymbol{x}'$ or, in other words, there is no $\boldsymbol{x}'$ that simultaneously decreases $\boldsymbol{y}_1(\boldsymbol{x}), \ldots, \boldsymbol{y}_m(\boldsymbol{x})$. Such $\boldsymbol{x}$ are also called *nondominated inputs*; analogously, the image $\boldsymbol{y}(\boldsymbol{x})$ of a nondominated $\boldsymbol{x}$ is sometimes referred to as a *nondominated output*. The set of all Pareto optimal points in $\mathcal{X}$ is referred to as the *Pareto Set*; denote the Pareto Set by $\mathbb{P}_{\mathcal{X}}$. The corresponding image of $\mathbb{P}_{\mathcal{X}}$ in $\mathcal{Y}$ is referred to as the *Pareto Front* and is denoted $\mathbb{P}_{\mathcal{Y}}$. Either of $\mathbb{P}_{\mathcal{Y}}$ and $\mathbb{P}_{\mathcal{X}}$ sets can be uncountable.

Example 6.3 (Pareto Set and Front). The so-called MOP2 function has a $d = 2$ dimensional input space $\mathcal{X} = [-2, 2]^2$ and $m = 2$ objective functions which are

$$\begin{aligned} y_1(x_1, x_2) &= 1 - \exp\left\{- \textstyle\sum_{i=1}^2 \left(x_i - \frac{1}{\sqrt{2}}\right)^2\right\} \\ y_2(x_1, x_2) &= 1 - \exp\left\{- \textstyle\sum_{i=1}^2 \left(x_i + \frac{1}{\sqrt{2}}\right)^2\right\}. \end{aligned} \tag{6.3.12}$$

Considered individually, each of $y_1(\boldsymbol{x})$ and $y_2(\boldsymbol{x})$ has global minimum equal to zero with minimizer $x_1 = 1/\sqrt{2} = x_2$ for $y_1(\boldsymbol{x})$ and $x_1 = -1/\sqrt{2} = x_2$ for $y_2(\boldsymbol{x})$. However, small $y_1(\boldsymbol{x})$ values correspond to large $y_2(\boldsymbol{x})$ values and vice versa. The Pareto Set, the collection of nondominated $\boldsymbol{x}$, can be shown to be the line segment

$$\mathcal{P}_{\mathcal{X}} = \left\{\boldsymbol{x} : -\frac{1}{\sqrt{2}} \leq x_1 = x_2 \leq \frac{1}{\sqrt{2}}\right\},$$

which is plotted as a black line within the green input space $\mathcal{X}$ shown in Fig. 6.7. The range space (Pareto Front) $\mathcal{Y}$ ($\mathbb{P}_{\mathcal{Y}}$) is the green area (black curve) shown in Fig. 6.8. ♦

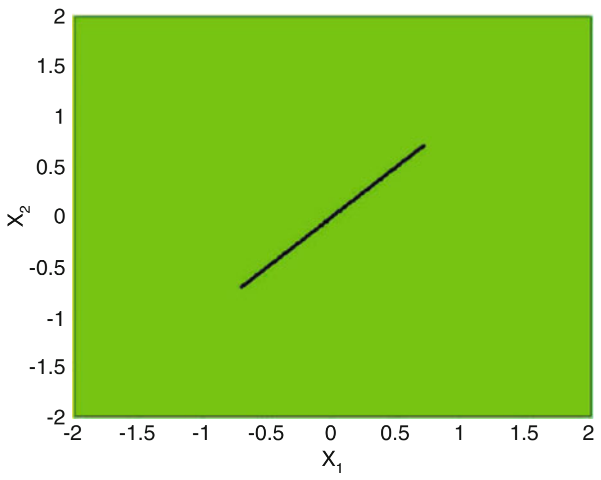


Fig. 6.7 $\mathcal{X}$ and the Pareto Set (black line segment) for the *MOP*2 function

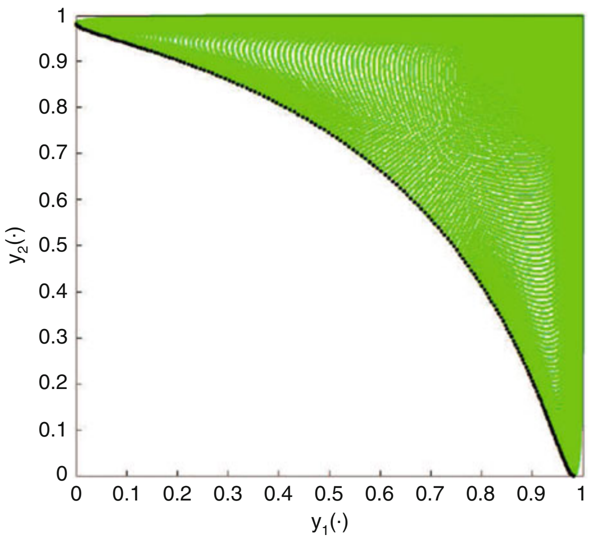


Fig. 6.8 $\mathcal{Y}$ and the Pareto Front (black curve) for the *MOP*2 function

The goal of this section is to show how "update rules" based on maximizing a *quality improvement* measure, denoted $QI(\boldsymbol{x})$, can be used to sequentially design a set of simulator runs to find a *Pareto optimum* of $\boldsymbol{y}(\boldsymbol{x})$. The expected improvement, introduced earlier, is one important quality improvement measure. More realistically, these methods allow a researcher to approximate $\mathbb{P}_{\mathcal{Y}}$ and $\mathbb{P}_{\mathcal{X}}$.

The improvement functions used for Pareto optimization are tailored to this problem. Among others Emmerich et al. (2006), Keane (2006), Forrester et al. (2007), Bautista (2009), and Svenson and Santner (2016) proposed heuristic improvement functions to guide the selection of new $\boldsymbol{x}$ inputs which are on or near the Pareto Set of the given problem. Before discussing specific improvement criteria, a general algorithm will be sketched that uses a generic $QI(\boldsymbol{x})$ for Pareto optimization of $\boldsymbol{y}(\boldsymbol{x}) = (y_1(\boldsymbol{x}), y_2(\boldsymbol{x}), \ldots, y_m(\boldsymbol{x}))^\top$ when $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \mathcal{X} \subset \mathbb{R}^d$.

6.3.6.1 Basic Pareto Optimization Algorithm

Assume that $\boldsymbol{y}(\boldsymbol{x})$ can be modeled as a draw from the process $\boldsymbol{Y}(\boldsymbol{x}) = (Y_1(\boldsymbol{x}), \ldots, Y_m(\boldsymbol{x}))^\top$; for technical simplicity many papers assume that the components of $\boldsymbol{Y}(\boldsymbol{x})$ are stochastically independent although this is not critical (see Svenson and Santner (2016) for an example using dependent $Y_i(\boldsymbol{x})$). The steps of the algorithm are as follows:

1. *Evaluate* $\boldsymbol{y}(\boldsymbol{x})$ *for inputs* $\boldsymbol{x}$ *from an initial space-filling design* $\mathcal{D}_n \subset \mathcal{X}$. *Let* $\boldsymbol{y}^{m,n} = \left[\boldsymbol{y}(\boldsymbol{x}_1) \cdots \boldsymbol{y}(\boldsymbol{x}_n)\right]$ *denote the* $m \times n$ *matrix whose* i^{th} *column is the* m *vector of outputs run at* $\boldsymbol{x}_i$, $i = 1, \ldots, n$.
2. *Estimate the unknown* $\boldsymbol{Y}(\boldsymbol{x})$ *process parameters based on* $\boldsymbol{y}^{m,n}$.
3. *Based on* $\boldsymbol{y}^{m,n}$, *calculate the Pareto Set, denoted* $\mathbb{P}^n_{\mathcal{X}}$, *and the Pareto Front, denoted* $\mathbb{P}^n_{\mathcal{Y}}$. *In words,* $\mathbb{P}^n_{\mathcal{X}}$ *is the set of nondominated inputs in* $\mathcal{D}_n$ *and* $\mathbb{P}^n_{\mathcal{Y}}$ *is the set of nondominated outputs in* $\boldsymbol{y}^{m,n}$.
4. *Find*
$$\boldsymbol{x}^{n+1} \in \max_{\boldsymbol{x}} QI(\boldsymbol{x}).$$
5. *Evaluate* $\boldsymbol{y}(\boldsymbol{x}^{n+1})$. *If the budget has been exhausted or a stopping criterion has been met, terminate the algorithm. Otherwise, repeat steps 2–5 with* $\boldsymbol{y}^{m,n+1} = \left[\boldsymbol{y}(\boldsymbol{x}_1) \cdots \boldsymbol{y}(\boldsymbol{x}_n)\ \boldsymbol{y}(\boldsymbol{x}_{n+1})\right]$.

This section will be completed by describing several methods that have been proposed in the literature for defining $QI(\boldsymbol{x})$ in terms of a *given* improvement function $I^*(y(\boldsymbol{x}))$. Then a selection of heuristically motivated $I^*(y(\boldsymbol{x}))$ will be listed.

- **Method 1—Expected Improvement:** Define the quality improvement associated with $I^*(\cdot)$ to be

$$QI_E(\boldsymbol{x}) = E\left[I^*(\boldsymbol{Y}(\boldsymbol{x})) \mid \boldsymbol{Y}^{m,n} = \boldsymbol{y}^{m,n}\right]; \tag{6.3.13}$$

  $QI_E(\boldsymbol{x})$ replaces the unknown $\boldsymbol{y}(\boldsymbol{x})$ by $\boldsymbol{Y}(\boldsymbol{x})$ and computes the expectation of $I^*(\boldsymbol{Y}(\boldsymbol{x}))$ with respect to the posterior distribution $[\boldsymbol{Y}(\boldsymbol{x}) \mid \boldsymbol{Y}^{m,n} = \boldsymbol{y}^{m,n}]$. $QI_E(\boldsymbol{x})$ is the natural extension of the expected improvement in (6.3.7).

- **Method 2—Improved Expectation:** Method 2 uses the quality indicator

$$QI_I(\boldsymbol{x}) = I^*\left(E\left[\boldsymbol{Y}(\boldsymbol{x}) \mid \boldsymbol{Y}^{m,n} = \boldsymbol{y}^{m,n}\right]\right), \tag{6.3.14}$$

which interchanges the expectation and $I^*(\boldsymbol{Y}(\boldsymbol{x}))$ in (6.3.13). Equation (6.3.14) is usually much simpler to compute than (6.3.13). The reader should note that Method 1 and Method 2 produce the same result if $I^*(\boldsymbol{y}(\boldsymbol{x}))$ is linear in $\boldsymbol{y}(\boldsymbol{x})$.

- **Method 3—Combined Expected Improvement and Improved Expectation:** Suppose that $I^*(\boldsymbol{y}(\boldsymbol{x}))$ is the form $G(\boldsymbol{y}(\boldsymbol{x})) \times H(\boldsymbol{y}(\boldsymbol{x}))$. Method 3 chooses the quality improvement indicator to be

$$QI_C(\boldsymbol{x}) = E\left[G(\boldsymbol{Y}(\boldsymbol{x})) \mid \boldsymbol{Y}^{m,n} = \boldsymbol{y}^{m,n}\right] \times H\left(E\left[\boldsymbol{Y}(\boldsymbol{x}) \mid \boldsymbol{Y}^{m,n} = \boldsymbol{y}^{m,n}\right]\right). \qquad (6.3.15)$$

  The quality improvement function $QI_C(\boldsymbol{x})$ has been proposed for use in cases where it is possible to compute the expectation of $G(\boldsymbol{Y}(\boldsymbol{x}))$, while that involving $H(\boldsymbol{Y}(\boldsymbol{x}))$ is "difficult." The formula (6.3.15) ignores any stochastic dependence between $G(\boldsymbol{Y}(\boldsymbol{x}))$ and $H(\boldsymbol{Y}(\boldsymbol{x}))$.

Formally, Method 3 generalizes both Methods 1 and 2 in that Method 3 becomes Method 1 if $H(\boldsymbol{y}(\boldsymbol{x})) = 1$, while Method 3 becomes Method 2 when $G(\boldsymbol{y}(\boldsymbol{x})) = 1$. As of the writing of this text, there is little theoretical basis for selecting among the three Methods to "best" find $\mathbb{P}_{\mathcal{X}}/\mathbb{P}_{\mathcal{Y}}$. However, there is much empirical evidence that shows improvement-based methods provide effective sequential designs for solving practical problems.

From a computational viewpoint, Method 1 is implemented by one of two schemes. Either one attempts to find an analytic expression for (6.3.13) as, for example, in (6.3.7). Alternatively, if analytic evaluation is not possible, one can turn to Monte Carlo methods by simulating a large number of draws from the distribution of $[\boldsymbol{Y}(\boldsymbol{x}) \mid \boldsymbol{Y}^{m,n} = \boldsymbol{y}^{m,n}]$, evaluating the improvement function at each draw, and averaging these values. The strong law of large numbers will guarantee convergence to (6.3.13) at any given $\boldsymbol{x}$.

Turning attention to improvement functions, Keane (2006) introduced the simplest of all improvement functions which is

$$I^*_K(\boldsymbol{y}(\boldsymbol{x})) = I\{\boldsymbol{y}(\boldsymbol{x}) \in R_n\},$$

where $I\{\cdot\}$ is the indicator function that takes the value 1 if $\boldsymbol{y}(\boldsymbol{x}) \in R_n$ and 0 otherwise and R_n denotes the region of the output space that is *not dominated by the current Pareto Front*, i.e.,

$$R_n = \left\{\boldsymbol{y}(\boldsymbol{x}) : \boldsymbol{x} \in \mathcal{X};\ \boldsymbol{y}(\boldsymbol{x}) \npreceq \boldsymbol{z},\ \forall\, \boldsymbol{z} \in \mathbb{P}^n_{\mathcal{Y}}\right\}.$$

The Method 1 improvement is trivial to calculate as

$$QI_K(\boldsymbol{x}) = P\left[\boldsymbol{Y}(\boldsymbol{x}) \in R_n \mid \boldsymbol{Y}^{m,n} = \boldsymbol{y}^{m,n}\right],$$

so that the method maximizes the probability of improving on the current Pareto Front as the update criterion. While simple to explain, the probability of improvement is not a very effective expected improvement function for Pareto optimization. Based on numerical studies, Keane (2006), Forrester et al. (2007), and Bautista

(2009) showed that using the probability of improvement tends to add outputs that are “clumped” together.

Keane (2006) also introduced the distance-weighted version

$$I^*_{KW}(\boldsymbol{y}(\boldsymbol{x})) = I\{\boldsymbol{y}(\boldsymbol{x}) \in R_n\} \times \min_{\boldsymbol{x}_i \in \mathbb{P}^n_{\mathcal{X}}} \sqrt{\sum_{k=1}^{m} (y_k(\boldsymbol{x}) - y_k(\boldsymbol{x}_i))^2}$$

of $I^*_K(\boldsymbol{y}(\boldsymbol{x}))$ with the goal of balancing exploration of the input space and exploitation of the surrogate approximation in the search for nondominated sets. Keane (2006) used Method 3 to form the quality improvement function

$$QI_{KW}(\boldsymbol{x}) = P\left[\boldsymbol{Y}(\boldsymbol{x}) \in R_n \mid \boldsymbol{Y}^{m,n}\right] \\ \times \min_{\boldsymbol{x}_i \in \mathbb{P}^n_{\mathcal{X}}} \sqrt{\sum_{k=1}^{m} \left(E\left[Y_k(\boldsymbol{x}) \mid \boldsymbol{Y}^{m,n} = \boldsymbol{y}^{m,n}\right] - y_k(\boldsymbol{x}_i)\right)^2}$$

for Pareto optimization. This quality improvement function can substantially outperform the probability of improvement, as the distance-based improvement criterion is larger for outputs that are farther from the current Pareto Front. Therefore, the magnitude of improvement is taken into consideration. This encourages the sequentially-added outputs to be more spread out in $\mathcal{Y}$ than when using the probability of improvement.

Svenson (2011) presented examples that show use of $QI_{KW}(\boldsymbol{y}(\boldsymbol{x}))$ can be less efficient than improvement functions based on the *maximin fitness function*, introduced by Balling (2003) to compare different finite sets of inputs using the $\succeq$ ordering in multi-objective function settings. The initial modification of fitness as an improvement function is

$$I^*_F(\boldsymbol{y}(\boldsymbol{x})) = -\max_{\boldsymbol{x}_i \in \mathbb{P}^n_{\mathcal{X}}} \min_{j=1,\ldots,m} \left(y_j(\boldsymbol{x}) - y_j(\boldsymbol{x}_i)\right) = \min_{\boldsymbol{x}_i \in \mathbb{P}^n_{\mathcal{X}}} \max_{j=1,\ldots,m} \left(y_j(\boldsymbol{x}_i) - y_j(\boldsymbol{x})\right) \quad (6.3.16)$$

proposed in Bautista (2009). That $I^*_F(\boldsymbol{y}(\boldsymbol{x}))$ is a heuristic choice of improvement function can be seen because if $I^*_F(\boldsymbol{y}(\boldsymbol{x})) > 0$, then $\boldsymbol{y}(\boldsymbol{x})$ is not dominated by any vectors in $\mathbb{P}^n_{\mathcal{Y}}$; if $I^*_F(\boldsymbol{y}(\boldsymbol{x})) < 0$, then $\boldsymbol{y}(\boldsymbol{x})$ is dominated by at least one vector in $\mathbb{P}^n_{\mathcal{Y}}$; and if $I^*_F(\boldsymbol{y}(\boldsymbol{x})) = 0$, then $\boldsymbol{y}(\boldsymbol{x})$ is weakly dominated by at least one vector in $\mathbb{P}^n_{\mathcal{Y}}$.

As discussed in Svenson and Santner (2016), a potential drawback of (6.3.16) is that $I^*_F(\boldsymbol{y}(\boldsymbol{x}))$ need not equal zero for $\boldsymbol{x}$ in the region dominated by the currently available outputs $\boldsymbol{y}(\boldsymbol{x}_i)$, $i = 1, \ldots, n$. Jones et al. (1998) and Schonlau (1997) used improvement functions which are zero in the currently dominated region; this feature accounts for their ability to provide both local and global criteria in the search for global optima.

Svenson and Santner (2016) modified $I^*_F(\boldsymbol{y}(\boldsymbol{x}))$ in (6.3.16) to be zero in the currently dominated region by proposing the use of

$$I^*_{TF}(\boldsymbol{y}(\boldsymbol{x})) = \min_{\boldsymbol{x}_i \in \mathbb{P}^n_{\mathcal{X}}} \max_{j=1,\ldots,m} \Big(y_j(\boldsymbol{x}_i) - y_j(\boldsymbol{x}) \Big)\; I\left\{ \min_{\boldsymbol{x}_i \in \mathbb{P}^n_{\mathcal{X}}} \max_{j=1,\ldots,m} \Big(y_j(\boldsymbol{x}_i) - y_j(\boldsymbol{x}) \Big) > 0 \right\}$$

which they call the *truncated maximin fitness function* improvement function. To select each new input $\boldsymbol{x}$, they maximize the Method 1 expected improvement

$$QI_E(\boldsymbol{x}) = E\left[I^*_{TF}(\boldsymbol{Y}(\boldsymbol{x})) \mid \boldsymbol{Y}^{m,n} = \boldsymbol{y}^{m,n} \right] \tag{6.3.17}$$

conditional on the previous function evaluations.

Example 6.3 *(Continued)*. While the Keane and distance-weighted Keane improvement functions are straightforward to understand, $I^*_{TF}(\boldsymbol{y}(\boldsymbol{x}))$ is more difficult to discern. Consider the MOP2 function (6.3.12) whose domain and output space are shown in Fig. 6.9. Suppose that $\boldsymbol{y}(\boldsymbol{x})$ has been calculated at the ten $\boldsymbol{x}$ values plotted in the left panel of Fig. 6.9; the corresponding $\boldsymbol{y}(\boldsymbol{x})$ outputs are shown in the right panel of the same figure. Recall that the true Pareto Set for the MOP2 function is the line segment shown in blue in the left panel of Fig. 6.9, while the corresponding true Pareto Front is the blue curve in the right panel. The Pareto Set and Front based (only) on the ten data values are the three red points in these same two panels; recall that the Pareto Front consists of those $\boldsymbol{y}(\boldsymbol{x}_j)$ that are not dominated by any other point in $\{\boldsymbol{y}(\boldsymbol{x}_i)\}^{10}_{i \neq j}$.

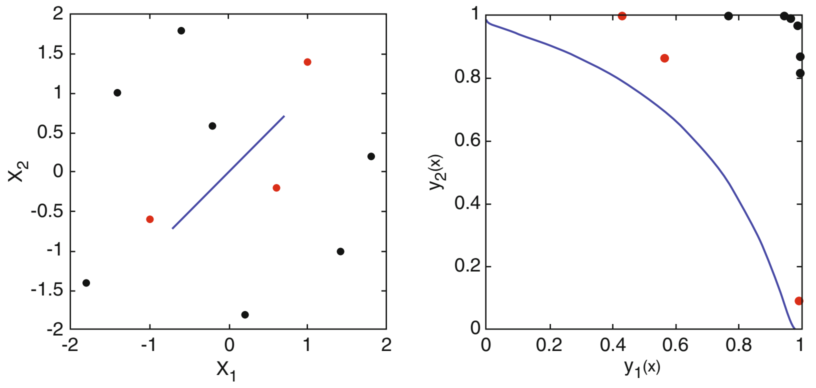


Fig. 6.9 Left panel: a design containing $n = 10$ inputs $\boldsymbol{x} = (x_1, x_2)$ in $[-2, +2]^2$ (black and red); the Pareto Set (in red) for the MOP2 function determined from the *ten design points*; the true Pareto Set for the MOP2 function (blue line). Right panel: the MOP2 function outputs $(y_1(\boldsymbol{x}), y_2(\boldsymbol{x}))$ corresponding to the ten-point design (black and red); the three $(y_1(\boldsymbol{x}), y_2(\boldsymbol{x}))$ pairs that form the Pareto Front (in red) determined from the *ten design points*; the true Pareto Front for the MOP2 function (blue curve)

Consider the first update to the ten-point design in Fig. 6.9 that is determined using the truncated maximin fitness function. Using the independent GP models for $y_1(\boldsymbol{x})$ and $y_2(\boldsymbol{x})$, calculation gives the Method 1 expected truncated maximin fitness function (6.3.17) that is plotted in Fig. 6.10. The maximum of $E[I^{*}_{TF}(\boldsymbol{Y}(\boldsymbol{x})) \mid \boldsymbol{Y}^{m,n} = \boldsymbol{y}^{m,n}]$ is 0.295 which occurs at $\boldsymbol{x} = (-0.1, -0.25)$. The augmentation of the $(y_1(\boldsymbol{x}), y_2(\boldsymbol{x}))$ scatterplot corresponding to adding $(-0.1, -0.25)$ to the design is shown as a red cross (+) in Fig. 6.11. The new $(y_1(\boldsymbol{x}), y_2(\boldsymbol{x}))$ point is "nearly" on the true Pareto Front and fills the void between the two right most Pareto Front points (in red) obtained from the ten point starting design. ♦

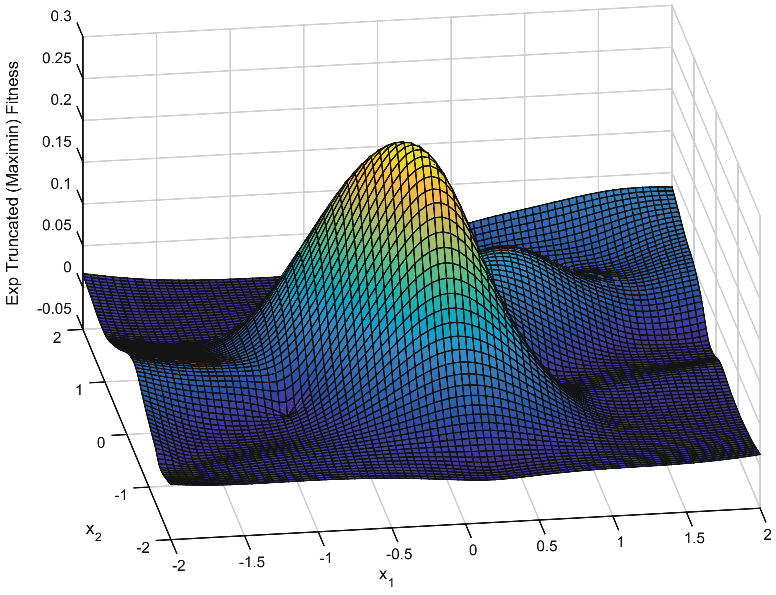


Fig. 6.10 Plot of $QI_E(x_1, x_2) = E[I^{*}_{TF}(\boldsymbol{Y}(x_1, x_2)) \mid \boldsymbol{Y}^{m,n} = \boldsymbol{y}^{m,n}]$ versus x_1 and x_2 for the ten-point design shown in Fig. 6.9

Other reasonable improvement functions are possible. For example, the "hypervolume indicator" improvement is discussed in the Chapter Notes in Sect. 6.5. Also see Coello Coello et al. (2006) for additional general discussion of Pareto optimization. Svenson and Santner (2016) discussed theoretical properties of several Pareto optimization improvement functions. Svenson and Santner (2016) and Svenson (2011) made empirical comparisons of the accuracy of estimated Pareto Fronts and Pareto Sets using the Basic Pareto Optimization Algorithm with different improvement functions and methods of constructing the associated quality indicator.

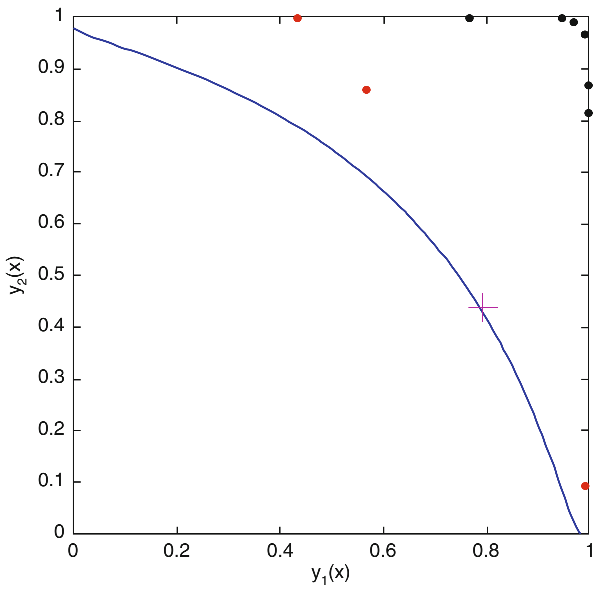


Fig. 6.11 Augmented $(y_1(\boldsymbol{x}), y_2(\boldsymbol{x}))$ plot of the MOP2 function shown as a red $+$ corresponding to adding a single $\boldsymbol{x}$ to the ten points shown in Fig. 6.9

6.4 Other Improvement Criterion-Based Designs

6.4.1 Introduction

Section 6.3 discussed sequential strategies for finding optima, in various senses, of simulator output. All the algorithms described in this section continue this theme by applying *heuristic* improvement criteria to form quality indicator functions which are used to update designs sequentially. For any specific goal, there may be many improvement criteria that are sensible. The next subsections describe how this improvement strategy has been implemented for problems of contour estimation, percentile estimation, and "global fit." In all three problems, the authors assume a continuous real-valued response $y(\boldsymbol{x})$ that can be described as a draw from a GP model $Y(\boldsymbol{x})$ with *constant mean*, say β, and specified parametric correlation function (see (2.2.3)). In practice, the power exponential or Gaussian correlation functions are often assumed. The emphasis in this section is on the improvement criteria and the algorithm for generating a design, rather than numerical implementation issues.

6.4.2 Contour Estimation

Ranjan et al. (2008) considered the goal of determining a contour of $y(\boldsymbol{x})$, i.e., of

$$\{\boldsymbol{x} \in \mathcal{X} : y(\boldsymbol{x}) = a\}\,, \tag{6.4.1}$$

where a is a user-specified constant and $y(\boldsymbol{x})$ is a simulator output in their methodology. Based on n training data outputs $\boldsymbol{y}^n$, let

$$\widehat{y}(\boldsymbol{x}_0) = \widehat{\beta} + \widehat{\boldsymbol{r}}_0^\top \widehat{\boldsymbol{R}}^{-1}\left(\boldsymbol{y}^n - \mathbf{1}_n \widehat{\beta}\right) \tag{6.4.2}$$

denote the MLE-EBLUP of $y(\boldsymbol{x}_0)$ and

$$s^2(\boldsymbol{x}_0) = \widehat{\sigma_Z^2}\left(1 - \widehat{\boldsymbol{r}}_0^\top \widehat{\boldsymbol{R}}^{-1} \widehat{\boldsymbol{r}}_0 + \frac{(1 - \mathbf{1}_n^\top \widehat{\boldsymbol{R}}^{-1} \widehat{\boldsymbol{r}}_0)^2}{\mathbf{1}_n^\top \widehat{\boldsymbol{R}}^{-1} \mathbf{1}_n}\right)$$

denote the estimated variance of (6.4.2). Here $\widehat{\beta}$ is the usual generalized least squares estimator of β; $\widehat{\boldsymbol{r}}_0$ is the estimate of the vector of correlations between $\boldsymbol{x}_0$ and the training data inputs; $\widehat{\boldsymbol{R}}$ is an estimate of the correlation matrix $\boldsymbol{R}$ of $\boldsymbol{Y}^n$, the $n \times 1$ model for $\boldsymbol{y}^n$; $\mathbf{1}_n$ is the $n \times 1$ vector of 1's; and $\widehat{\sigma_Z^2}$ is the MLE of σ_Z^2 (see (3.2.7) and (3.2.8)).

Ranjan et al. (2008) proposed the use of a probability-based improvement function built from

$$\begin{aligned} i(\boldsymbol{x}) &= \alpha^2 s^2(\boldsymbol{x}) - \min\left\{(y(\boldsymbol{x}) - a)^2, \alpha^2 s^2(\boldsymbol{x})\right\} \\ &= \begin{cases} \alpha^2 s^2(\boldsymbol{x}) - (y(\boldsymbol{x}) - a)^2, & \text{if } y(\boldsymbol{x}) \in (a - \alpha\, s(\boldsymbol{x}),\ a + \alpha\, s(\boldsymbol{x})) \\ 0 & , \text{otherwise} \end{cases} \end{aligned} \tag{6.4.3}$$

where $\alpha > 0$ is a constant used to tune the sequential update method alluded to below (see also Roy and Notz (2014)). It is easy to see that the maximum possible $i(\boldsymbol{x})$ improvement is $\alpha^2 s^2(\boldsymbol{x})$. Equation (6.4.3) suggests that the improvement at $\boldsymbol{x}$ has a "confidence interval" interpretation. An improvement occurs if the interval $y(\boldsymbol{x}) \pm \alpha\, s(\boldsymbol{x})$ includes a.

The probability-based improvement function obtained from $i(\boldsymbol{x})$ is

$$I(\boldsymbol{x}) = \alpha^2 S^2(\boldsymbol{x}) - \min\left\{(Y(\boldsymbol{x}) - a)^2, \alpha^2 S^2(\boldsymbol{x})\right\}$$

where $S^2(\boldsymbol{x})$ is defined in terms of the random variables $Y(\boldsymbol{x})$ and $\boldsymbol{Y}^n$. The conditional expected $I(\boldsymbol{x})$ given the training data is used to sequentially add design points $\boldsymbol{x}$ to an existing design. Using the fact that the conditional distribution of $Y(\boldsymbol{x})$ given the training data $\boldsymbol{Y}^n$ at stage n is approximately $N\left(\widehat{y}(\boldsymbol{x}), s^2(\boldsymbol{x})\right)$, Ranjan et al. (2008) adopted Method 1 expected improvement to form the quality indicator

$$\begin{aligned} E\left[I(Y(\boldsymbol{x})) \mid \boldsymbol{Y}^n\right] = & \left[(\alpha s(\boldsymbol{x}))^2 - (\widehat{y}(\boldsymbol{x}) - a)^2\right]\left[\Phi\left(\frac{a-\widehat{y}(\boldsymbol{x})}{s(\boldsymbol{x})} + \alpha\right) - \Phi\left(\frac{a-\widehat{y}(\boldsymbol{x})}{s(\boldsymbol{x})} - \alpha\right)\right] \\ & + 2(\widehat{y}(\boldsymbol{x}) - a)\, s^2(\boldsymbol{x})\left[\phi\left(\frac{a-\widehat{y}(\boldsymbol{x})}{s(\boldsymbol{x})} + \alpha\right) - \phi\left(\frac{a-\widehat{y}(\boldsymbol{x})}{s(\boldsymbol{x})} - \alpha\right)\right] \\ & - \int_{a-\alpha s(\boldsymbol{x})}^{a+\alpha s(\boldsymbol{x})} (y - \widehat{y}(\boldsymbol{x}))^2\, \phi\left(\frac{y-\widehat{y}(\boldsymbol{x})}{s(\boldsymbol{x})}\right) dy\,, \end{aligned} \tag{6.4.4}$$

where $\Phi(\cdot)$ and $\phi(\cdot)$ denote the cumulative distribution function and probability density function of the standard normal distribution, respectively.

Although not immediately obvious, it turns out that the expected improvement (6.4.4) has both a local search component and a global search component. The constant α determines the extent to which the search is local versus global. Larger values of α produce a more global search. The greatest improvement is achieved for points $\boldsymbol{x}^*$ where $\widehat{y}(\boldsymbol{x}^*)$ is close to a and $s(\boldsymbol{x}^*)$ is large. The examples in Ranjan et al. (2008) showed the effectiveness of using (6.4.4) to guide the sequential selection of inputs to identify the contour (6.4.1). As anticipated, their algorithm adds points that are near the true contour (local search) and occasionally points that are in regions of high uncertainty (global search).

6.4.3 Percentile Estimation

Roy and Notz (2014) extended the results of Ranjan et al. (2008) to estimate percentiles of $y(\boldsymbol{x})$ when the inputs have a known distribution that is denoted by $\boldsymbol{X}$. As motivation, recall Example 1.3 which describes a simulator for the flow of water, $y(\boldsymbol{x})$, through a borehole with eight characteristics $\boldsymbol{x}$. Suppose that the simulator inputs are not known exactly but only up to a distribution that specifies their uncertainty. Let $\boldsymbol{X}$ denote the random inputs having this uncertainty distribution. Then the induced distribution $y(\boldsymbol{X})$ describes the uncertainty in the water flow. In fact, when the actual range of values of the output $y(\boldsymbol{x})$ is not well known, it may not be possible to specify what one means by a "large" value of $y(\boldsymbol{x})$ other than by specifying a large percentile.

Given p, $0 < p < 1$, the goal of Roy and Notz (2014) was to estimate the p^{th} percentile, ζ_p, of the $y(\boldsymbol{X})$ distribution, i.e.,

$$P\left[y(\boldsymbol{X}) \le \zeta_p\right] = p\,.$$

For simplicity, Roy and Notz (2014) assumed that $\boldsymbol{X}$ is uniformly distributed over $\mathcal{X}$, but their results can be easily modified to handle any known continuous distribution. In the following discussion, the same model and the same notation as in Sect. 6.4.2 are used.

Roy and Notz (2014) discussed two types of improvement criteria. One is a confidence interval-inspired approach which is motivated as follows. Given there is a fair idea about the location of the p^{th} percentile ζ_p, one should look at inputs for which the output lies in an interval about ζ_p to refine our estimate of ζ_p. The other approach is a hypothesis testing-inspired approach; one selects design points at which the response is not "significantly different" from ζ_p. For either approach, Roy and Notz (2014) used the following algorithm:

1. *Generate a large random sample from the distribution of the input variables. (They use a large maximin distance Latin hypercube sample in their examples.)*
2. *Use the EBLUP based on the training data to predict the values of the output for this large sample.*
3. *Numerically estimate ζ_p from the ordered vector of predicted values obtained in Step 2.*

6.4.3.1 Approach 1: A Confidence Interval-Based Criterion

Given the current estimate of the p^{th} percentile value, ζ_p, of the induced distribution of the output variable, define the theoretical improvement at any untried $\boldsymbol{x}$ to be

$$i(\boldsymbol{x}) = \begin{cases} h\left(y(\boldsymbol{x}) - \zeta_p, 1/s(\boldsymbol{x})\right), & \text{if } y(\boldsymbol{x}) \in (\zeta_p - \alpha s(\boldsymbol{x}),\ \zeta_p + \alpha s(\boldsymbol{x})) \\ \quad 0 & , \text{ otherwise} \end{cases}, \tag{6.4.5}$$

where $\alpha > 0$, and $h(w, v)$ is a decreasing function of $|w|$ and v. According to (6.4.5), if the current estimate of the p^{th} percentile, ζ_p, lies within $\alpha s(\boldsymbol{x})$ units of the response at input site $\boldsymbol{x}$, the improvement at that design point is set equal to $h(y(\boldsymbol{x}) - \zeta_p, 1/s(\boldsymbol{x}))$. If not the improvement is set equal to 0. The corresponding probability-based improvement function, $I(\boldsymbol{x})$, is obtained by replacing $y(\boldsymbol{x})$ with $Y(\boldsymbol{x})$.

As an example, suppose

$$h\left(y(\boldsymbol{x}) - \zeta_p, 1/s(\boldsymbol{x})\right) = (\alpha s(\boldsymbol{x}))^g - (y(\boldsymbol{x}) - \zeta_p)^g ,$$

where g is a positive even integer. The theoretical improvement (6.4.5) becomes

$$i(\boldsymbol{x}) = \begin{cases} (\alpha s(\boldsymbol{x}))^g - (y(\boldsymbol{x}) - \zeta_p)^g, & \text{if } y(\boldsymbol{x}) \in (\zeta_p - \alpha s(\boldsymbol{x}),\ \zeta_p + \alpha s(\boldsymbol{x})) \\ \quad 0 & , \text{ otherwise} \end{cases}, \tag{6.4.6}$$

and the corresponding probability-based improvement function, $I_g(\boldsymbol{x})$, is obtained by replacing $y(\boldsymbol{x})$ with $Y(\boldsymbol{x})$. Formally, the improvement (6.4.3) of Ranjan et al. (2008) is a special case of (6.4.6) when $\zeta_p = a$ and $g = 2$. Roy and Notz (2014) showed that the relative amounts of local versus global search in their probability-based improvement criterion $I_g(\boldsymbol{x})$ can be controlled by changing g and α. From (6.4.6) it is clear that, at any iteration, increasing α results in a wider interval of candidate design points (leading to a more global search) from which one may pick the most "informative." The use of g to control the relative amount of local versus global search was inspired by Schonlau et al. (1998).

The Method 1 expected improvement quality indicator (6.3.13) corresponding to (6.4.6) is derived in Roy and Notz (2014). For example, when $g = 2$

$$\begin{aligned} E\left[I(\boldsymbol{x}) \mid \boldsymbol{Y}^n\right] = &\left[(\alpha s(\boldsymbol{x}))^2 - (\widehat{y}(\boldsymbol{x}) - \zeta_p)^2\right] \\ &\times \left[\Phi\left(\frac{\zeta_p - \widehat{y}(\boldsymbol{x})}{s(\boldsymbol{x})} + \alpha\right) - \Phi\left(\frac{\zeta_p - \widehat{y}(\boldsymbol{x})}{s(\boldsymbol{x})} - \alpha\right)\right] \\ &+ 2(\widehat{y}(\boldsymbol{x}) - \zeta_p)\, s^2(\boldsymbol{x}) \left[\phi\left(\frac{\zeta_p - \widehat{y}(\boldsymbol{x})}{s(\boldsymbol{x})} + \alpha\right) - \phi\left(\frac{\zeta_p - \widehat{y}(\boldsymbol{x})}{s(\boldsymbol{x})} - \alpha\right)\right] \\ &- \int_{\zeta_p - \alpha s(\boldsymbol{x})}^{\zeta_p + \alpha s(\boldsymbol{x})} (y - \widehat{y}(\boldsymbol{x}))^2\, \phi\left(\frac{y - \widehat{y}(\boldsymbol{x})}{s(\boldsymbol{x})}\right) dy\,. \end{aligned}$$

6.4.3.2 Approach 2: A Hypothesis Testing-Based Criterion

A hypothesis testing inspired improvement function can be derived from the "theoretical" *discrepancy*

$$d_\epsilon(\boldsymbol{x}) = \begin{cases} \dfrac{\left(y(\boldsymbol{x}) - \zeta_p\right)^2 + \epsilon}{s^2(\boldsymbol{x})}, & \text{if } s(\boldsymbol{x}) \neq 0 \\ \infty & , \text{otherwise} \end{cases} \tag{6.4.7}$$

between the current estimate of the p^{th} percentile, ζ_p, and $y(\boldsymbol{x})$ where $\epsilon > 0$. The corresponding probability-based discrepancy is

$$D_\epsilon(\boldsymbol{x}) = \begin{cases} \dfrac{\left(Y(\boldsymbol{x}) - \zeta_p\right)^2 + \epsilon}{s^2(\boldsymbol{x})}, & \text{if } s(\boldsymbol{x}) \neq 0 \\ \infty & , \text{otherwise} \end{cases}.$$

The idea is to choose that design site at which the response is not significantly different from the p^{th} percentile, ζ_p. The negative (or reciprocal) of $D_\epsilon(\boldsymbol{x})$ is an improvement function.

The intuition behind this discrepancy is as follows. First, if a particular design site has already been observed, the mean squared prediction error, $s^2(\boldsymbol{x})$, at that point is zero; hence the discrepancy is set equal to infinity. Second, suppose there are two competing design sites $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ and the responses at both $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are equal to the current estimate of the p^{th} percentile, ζ_p. The value of the theoretical discrepancy in Eq. (6.4.7) without the ϵ term is *zero* for *both* $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ and thus does not consider the associated uncertainties $s^2(\boldsymbol{x}_1)$ and $s^2(\boldsymbol{x}_2)$. Including ϵ results in nonzero values of the theoretical discrepancy; that value $\boldsymbol{x}_1$ or $\boldsymbol{x}_2$ having the larger $s^2(\boldsymbol{x})$ will have smaller value of the theoretical discrepancy. Thus, selecting the design site with the smallest value of the theoretical discrepancy ensures that the design site with the highest mean squared prediction error is selected. Increasing the value of ϵ leads to a more global search. As discussed above, the inclusion of ϵ ensures that design sites with higher values of $s^2(\boldsymbol{x})$ are selected over other candidate sites.

Roy and Notz (2014) proposed a quality indicator based on a Method 1 expected discrepancy. They used examples to demonstrate the effectiveness of their method and to emphasize the need for a sufficiently large initial sample size in its implementation.

6.4.4 Global Fit

A very challenging problem is developing criteria for the goal of producing a predictor of the simulator output that has "good" overall fit, i.e., gives "good" predictions over the entire input space, $\mathcal{X}$.

Lam and Notz (2008) proposed the following probability-based improvement function to provide good overall fit. For each $\boldsymbol{x} \in \mathcal{X}$, set

$$I(\boldsymbol{x}) = (Y(\boldsymbol{x}) - y(\boldsymbol{x}_{j^*}))^2 \,,$$

where $\boldsymbol{x}_{j^*}$ is the training input *closest* (in Euclidean distance) to $\boldsymbol{x}$. Hence $y(\boldsymbol{x}_{j^*})$ is known. The intuition behind this criterion is to place high value on "informative" regions in the domain that will help improve the global fit of the model. By informative Lam and Notz (2008) mean regions with significant error in the response values.

The Method 1 expected improvement for this probability-based improvement function is

$$E\left[I(Y(\boldsymbol{x})) \mid \boldsymbol{Y}^n\right] = \left(\widehat{y}(\boldsymbol{x}) - y(\boldsymbol{x}_{j^*})\right)^2 + s^2(\boldsymbol{x})\,. \tag{6.4.8}$$

The expected improvement in (6.4.8) consists of two search components—local and global. The local component of $E[I(y(\boldsymbol{x}))]$ is large at $\boldsymbol{x}$ where $\widehat{y}(\boldsymbol{x})$ has the largest increase over its nearest sampled point. The global component is large for points with the largest $s^2(\boldsymbol{x})$, i.e., points about which there is large uncertainty that tend to be far from existing sampled points.

Lam and Notz (2008) compared the performance of this criterion with sequential and fixed-sample implementations of design criteria based on mean squared prediction error, changes in measures based on cross-validation error, entropy, and a space-filling design. Their method performs well on response surfaces that look "nonstationary," by which they mean surfaces that have regions of high volatility (appear very rough) as well as regions that are very flat. It performs satisfactorily in examples where the response surface looks like the realization of a stationary GP.

Loeppky et al. (2010) investigated strategies for adding design points in *batches* to improve the global fit of the GP model. They refer to this as "emulator maturity," the idea being that one seeks to improve the emulator (predictor) of the physical process by adding observations from the computer simulation (and, perhaps, from the physical process itself). They consider several strategies. One is the one-at-a-time sequential strategy using the expected improvement criterion of Lam and Notz

(2008). Others are batch sequential implementations of the entropy criterion, integrated mean squared prediction error, maximum mean squared prediction error, and two proposed distance-based strategies that add points in batches to maintain some degree of "space-fillingness." Also considered is a one-stage symmetric LHD that takes a total number of observations equal to the total taken by the batch strategies. The surprising result is that in the examples considered by Loeppky et al. (2010), the one-stage symmetric LHD is never outperformed in controlling average prediction errors and rarely outperformed in controlling maximum prediction errors. The reason this is surprising is that many of the batch strategies make use of the observations from the first stage, while the one-stage design does not. It seems counterintuitive that ignoring information that determines the predictor should lead to improved performance. The reasons are unclear, but the following may be relevant.

First, it may be that there are better improvement criteria and batch strategies for obtaining good global fit than those studied by Loeppky et al. (2010). If so, these "better strategies" might at least be competitive with the one-stage design. Second, the test function used in the study varies smoothly with each input and has most of its output variation explained by relatively simple effects. Thus, the (stationary) GP model appears to be a reasonable model for the test function. As discussed in Sect. 5.1.2, assuming the GP model is correct, for sufficiently large sample sizes, no design will outperform space-filling designs (space-filling in the sense that as the sample size increases, the design becomes dense in $\mathcal{X}$) in terms of the rate at which the maximum of the mean squared prediction error decreases as n_s increases. So to the extent that the output from the computer simulator resembles a realization of a GP model, space-filling designs should perform well in terms of producing predictors with good overall global fit for large sample sizes.

Less clear is the possible performance of various design strategies when the simulator output does not look like the realization of a (stationary) GP model. EBLUPs based on the GP model are interpolators, so EBLUPs will often provide good global fit with enough data. Whether theoretical results about the asymptotic performance of space-filling designs for the GP model are valid when the true response does not resemble a realization of a GP model is not clear. And to what extent asymptotic results provide insight into performance for smaller sample sizes is also not clear. More research and more extensive testing on a variety of response surfaces with differing features (some that appear to be realizations of a GP model, some that have distinctly nonstationary features) might improve understanding of the behavior of different design criteria.

6.5 Chapter Notes

While the references in the individual sections provide additional details concerning the procedures discussed in that section, Notz (2015) gives an overview of additional methods based on expected improvement.

6.5.1 The Hypervolume Indicator for Approximations to Pareto Fronts

Several quality measures have been introduced for comparing different approximations to a given Pareto Front, $\mathbb{P}_{\mathcal{Y}}$, and also to provide improvement functions $I(\boldsymbol{x})$. This subsection will illustrate these ideas using one such measure called the "*hypervolume indicator*"; Zitzler et al. (2008) gives a detailed survey of this topic and other quality measures. Below, the hypervolume indicator will be defined, and then two applications of this indicator will be described.

To define the hypervolume indicator, one must first extend the idea of (weak) dominance of one *point* in a given set $\mathcal{Y}$ by another *point* in $\mathcal{Y}$ (see Sect. 6.3.6) to one *subset* of points in $\mathcal{Y}$ by another *subset* of points in $\mathcal{Y}$. Suppose that $\boldsymbol{A}$ and $\boldsymbol{B}$ are subsets of $\mathcal{Y}$; $\boldsymbol{A}$ is said to (*weakly*) *dominate* $\boldsymbol{B}$, denoted $A \succ B$ ($A \succeq B$), if *every* point in $\boldsymbol{B}$ is dominated by one or more points in $\boldsymbol{A}$. Now consider $\boldsymbol{A} \subset \mathcal{Y}$ and fix a reference point $\boldsymbol{v}$ which is weakly dominated by every $\boldsymbol{y} \in \mathcal{Y}$. The hypervolume indicator of $\boldsymbol{A}$ is defined to be

$$I_H(\boldsymbol{A}, \boldsymbol{v}) = \int_{\mathcal{Y}} I\{\boldsymbol{y} \mid \boldsymbol{y} \succeq \boldsymbol{v},\ \boldsymbol{A} \succeq \{\boldsymbol{y}\}\}\, d\boldsymbol{y}$$

where $I\{\cdot\}$ is the indicator function. In words, $I_H(\boldsymbol{A}, \boldsymbol{v})$ is the volume of the set of points in $\mathcal{Y}$ that are dominated by one or more points from $\boldsymbol{A}$ but also dominate the reference point $\boldsymbol{v}$. As an example, the green area of Fig. 6.12 illustrates $I_H(\boldsymbol{A}, \boldsymbol{v})$ for a finite set of points in 2-D.

The most important property of the hypervolume indicator, used to compare approximations to a given Pareto Front, is its *strict monotonicity* with respect to dominance. This means that for sets $\boldsymbol{A}^1$ and $\boldsymbol{A}^2$ with $\boldsymbol{A}^1 \succ \boldsymbol{A}^2$, then $I_H(\boldsymbol{A}^1, \boldsymbol{v}) > I_H(\boldsymbol{A}^2, \boldsymbol{v})$.

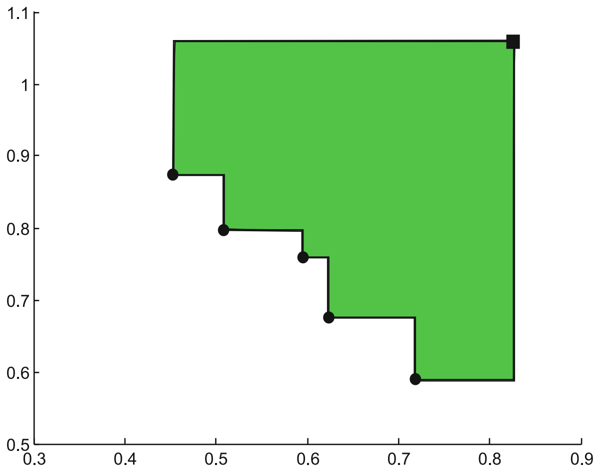


Fig. 6.12 The filled circles form a five-point set $\boldsymbol{A}$; the filled square is the reference point $\boldsymbol{v}$; and the shaded region is $I_H(\boldsymbol{A}, \boldsymbol{v})$

While there are alternate quality measures, say $I(\boldsymbol{A})$, that have the weak monotonicity property, i.e., $I(\boldsymbol{A}^1) \geq I(\boldsymbol{A}^2)$ whenever $\boldsymbol{A}^1 \geq \boldsymbol{A}^2$, the hypervolume indicator and a related indicator, the weighted hypervolume indicator, are the only known quality measures that are *strictly* monotonic (Zitzler et al. (2008)).

Despite its strict monotonicity, the hypervolume indicator has three disadvantages when used as a quality measure for comparing approximations to a Pareto Set. First, the value of $I_H(\boldsymbol{A}, \boldsymbol{v})$ depends on the *scaling* of the outputs. Second, $I_H(\boldsymbol{A}, \boldsymbol{v})$ requires the user *know* an upper bound $\boldsymbol{v}$ on the output space. And, lastly, $I_H(\boldsymbol{A}, \boldsymbol{v})$ is computationally *expensive*; according to Fonseca et al. (2006), the best known algorithms for calculating the hypervolume indicator have running times that are exponential in m, the number of components of $\boldsymbol{y}(\boldsymbol{x})$. All three of these disadvantages are addressed by papers that use the hypervolume indicator for comparisons of regions or to define an improvement criterion.

Turning attention to comparing approximations to a Pareto Front, suppose that two approximations have been formed by determining the nondominated simulator evaluations, $\boldsymbol{y}(\boldsymbol{x})$, that have been determined by two different methods of estimating the Pareto Front. Because the estimated Fronts are comprised of function evaluations, they must necessarily have smaller hypervolume indicators than the true Pareto Front. The approximation having the *larger* value of the hypervolume is considered better.

The second application of the hypervolume indicator is to define an improvement criterion. Emmerich et al. (2006) defined the *hypervolume improvement criterion* as

$$I_{\mathbb{H}}(\mathbf{y}(\boldsymbol{x})) = \begin{cases} 0, & \boldsymbol{y}(\boldsymbol{x}) \preceq \mathbb{P}^n_{\mathcal{Y}} \text{ or } \boldsymbol{y}(\boldsymbol{x}) \not\leq \boldsymbol{v} \\ I_H\left(\{\boldsymbol{y}(\boldsymbol{x})\} \cup \mathbb{P}^n_{\mathcal{Y}}, \boldsymbol{v}\right) - I_H\left(\mathbb{P}^n_{\mathcal{Y}}, \boldsymbol{v}\right), & \text{otherwise} \end{cases} \tag{6.5.1}$$

for a suitable reference point $\boldsymbol{v}$. Then the next point in the sequential design is obtained by maximizing the expected value of a probabilistic version of (6.5.1).

6.5.2 Other MSPE-Based Optimal Designs

Leatherman et al. (2018) discussed designs that minimize a weighted version of the IMSPE criterion and hence can be viewed as Bayesian designs. Let $\text{MSPE}_{\boldsymbol{\kappa}}[\widehat{y}_{\boldsymbol{\kappa}}(\boldsymbol{x})]$ be an extended version of the MSPE notation (6.2.4) which emphasizes that $\boldsymbol{\kappa}$ is the correlation under which both the EBLUP $\widehat{y}_{\boldsymbol{\kappa}}(\boldsymbol{x})$ and the MSPE are calculated. The Leatherman et al. (2018) criterion chooses $\mathcal{D}$ to minimize

$$WI(\mathcal{D}) = \int \left(\int_{\mathcal{X}} \text{MSPE}_{\boldsymbol{\kappa}}[\widehat{y}_{\boldsymbol{\kappa}}(\boldsymbol{x})]\, d\boldsymbol{x} \right) \pi(\boldsymbol{\kappa})\, d\boldsymbol{\kappa},$$

with respect to a weight $\pi(\boldsymbol{\kappa})$ on $\boldsymbol{\kappa}$ ($\pi(\boldsymbol{\kappa})$ is a prior if one is Bayesian). An n-point design $\mathcal{D}_{wimpse}$ is *WIMSPE-optimal* if it minimizes $WI(\mathcal{D})$. WIMSPE-optimal designs can be extremely time-consuming to construct. While WIMSPE-optimal designs can provide smaller EMSPE prediction for test bed functions than designs which are IMSPE-optimal against a *single* $\boldsymbol{\kappa}$, the examples of Leatherman et al. (2018) suggested that the magnitude of the improvements are often not worth the computation cost of determining the design.

In a similar spirit, Leatherman et al. (2017) recommended specific IMSPE-optimal designs for calibrated prediction of the mean of a physical process based on combined physical system observations and simulator output.

Weaver et al. (2016) developed an approach to optimizing computationally expensive Bayesian design criterion functions such as $WI(\mathcal{D})$. Let $\Lambda(\mathcal{D})$ denote an arbitrary design criterion function, typically represented as an integral of a utility (or loss) function with respect to the joint distribution of uncertain parameters and unobserved future data. Evaluation of $\Lambda(\mathcal{D})$ for a single proposed design $\mathcal{D}$ may involve both Monte Carlo simulation and MCMC, both computationally intensive procedures indicating that a limited budget of design criterion realizations will be available for design optimization. The method involves taking a portion of this budget to build an initial GP surrogate for the stochastic $\Lambda(\mathcal{D})$ and then invoking the EQI of Picheny et al. (2013) to sequentially optimize $\Lambda(\cdot)$ over designs $\mathcal{D}$ using the remainder of the budget.

6.5.3 Software for Constructing Criterion-Based Designs

The designs discussed in this chapter must be produced numerically. Below is a partial list of software packages that will generate these designs.

1. JMP 12 will produce maximum entropy designs and IMSPE-optimal designs in the space-filling design option under the DOE menu. The Gaussian correlation function is assumed. For both designs, the user must specify the values of the correlation parameters, $\boldsymbol{\xi}$.
2. Leatherman gives MATLAB code for finding IMSPE- and weighted IMSPE-optimal designs (https://www2.kenyon.edu/Depts/Math/Leatherman/CompExpDesgs_Pred/). The software constructs the design under the assumption that all data come from a stationary Gaussian process with Gaussian correlation function. For IMSPE-optimal designs, the user must specify the values of the correlation parameters. For weighted IMSPE-optimal designs, the user must specify a beta distribution to weight the correlation parameters.
3. The DiceDesign package in R will generate maximum entropy designs given values for the correlation parameters for a correlation matrix based on a spherical variogram. A description of the DiceDesign package is available at http://cran.r-project.org/web/packages/DiceDesign/DiceDesign.pdf
4. Dakota is a software package developed at Sandia National Laboratories for the analysis of data from predictive simulations. The package allows one to implement the sequential design strategies of Schonlau et al. (1998) and Jones et al. (1998), sometimes referred to as efficient global optimization (EGO). Dakota

can be downloaded from http://dakota.sandia.gov. This package also implements a variety of other optimization methods.

5. SPACE (Stochastic Process Analysis of Computer Experiments) is code written by Matthias Schonlau that also implements the strategies of Schonlau et al. (1998) and Jones et al. (1998). The code can be found at http://www.schonlau.net/space.html.
6. The max_EQI function distributed with the R package DiceOptim utilizes an evolutionary search algorithm with a Newton or quasi-Newton optimization method (see Appendix C) to maximize the EQI function of Picheny et al. (2013). A description of the DiceOptim package is available at http://cran.r-project.org/web/packages/DiceOptim/DiceOptim.pdf
7. The laGP package in R implements blackbox constrained optimization via augmented Lagrangians as described in Gramacy et al. (2016), in addition to other tasks involving inference with GPs. A description of the laGP package is available at http://bobby.gramacy.com/r_packages/laGP
8. Svenson (2011) describes MATLAB code that implements sequential designs for Pareto optimization.

Chapter 7
Sensitivity Analysis and Variable Screening

7.1 Introduction

This chapter discusses sensitivity analysis and the related topic of variable screening. The setup is as follows. A vector of inputs $\boldsymbol{x} = (x_1, \ldots, x_d)$ is given which potentially affects a "response" function $y(\boldsymbol{x}) = y(x_1, \ldots, x_d)$. Sensitivity analysis seeks to quantify how variation in $y(\boldsymbol{x})$ can be apportioned to the inputs $x_1, \ldots, x_d$ *and to the interactions among these inputs*. Variable selection is more decision oriented in that it seeks to simply determine, for each input, whether that input is "active" or not. However, the two notions are related and variable screening procedures use some form of sensitivity analysis to assess the activity of each candidate input. Hence sensitivity analysis will be described first and then, using sensitivity analysis (SA) tools, two approaches to variable selection will be presented.

To fix ideas concerning sensitivity analysis, consider the function

$$y(x_1, x_2) = x_1 + x_2. \tag{7.1.1}$$

with domain $(x_1, x_2) \in (0, 1) \times (0, 2)$. One form of sensitivity analysis is based on examining the *local change* in $y(\boldsymbol{x})$ as x_1 or x_2 increases by a small amount starting from (x_1^0, x_2^0). This change can be determined from the partial derivatives of $y(\cdot)$ with respect to x_1 and x_2; in this example,

$$\frac{\partial y(x_1, x_2)}{\partial x_1} = 1 = \frac{\partial y(x_1, x_2)}{\partial x_2},$$

so that we can assert that small changes in the inputs parallel to the x_1 or the x_2 axes *starting from any input* have the same effect on $y(\cdot)$.

A more *global assessment* of the sensitivity of $y(\boldsymbol{x})$ with respect to any component x_i, $i = 1, \ldots, d$, examines the change in $y(\boldsymbol{x})$ as x_i ranges over its domain for *fixed* values of the remaining inputs. In the case of (7.1.1), for fixed x_1^0 it is easy to see that the range of $y(x_1^0, x_2)$ as x_2 varies over $(0, 2)$, is $2 = \max_{x_2} y(x_1^0, x_2) - \min_{x_2} y(x_1^0, x_2) = y(x_1^0, 2) - y(x_1^0, 0)$ which is *twice* as large as

$1 = \max_{x_1} y(x_1, x_2^0) - \min_{x_1} y(x_1, x_2^0)$, the range of $y(x_1, x_2^0)$ over x_1 for any fixed x_2^0. Thus this second method of assessing sensitivity concludes that $y(x_1, x_2)$ is twice as sensitive to x_2 as x_1.

This example illustrates two of the approaches that have been used to assess the influence of inputs on a given output. *Local sensitivity analysis* measures the change in the slope of the tangent to $y(\boldsymbol{x})$ at $\boldsymbol{x}$ in the direction of a given input axis j, fixing the remaining inputs. *Global sensitivity analysis* measures the change in $y(\boldsymbol{x})$ as one (or more inputs) vary over their entire *domain* when the remaining inputs are fixed. As the example above shows, the different criteria can lead to different conclusions about the sensitivity of $y(\boldsymbol{x})$ to its inputs. When it is determined that certain inputs have relatively *little* effect on the output, we can set these inputs to nominal values and reduce the dimensionality of the problem allowing us to perform a more exhaustive investigation of a predictive model with a fixed budget for runs.

Sensitivity analysis is also useful for identifying interactions between variables. When interactions *do not exist*, the effect of any given input is the same regardless of the values of the other inputs; in this case, the relationship between the output and inputs is said to be *additive* and is readily understandable (Fig. 7.1). When interactions exist, the effects of some inputs on the output will depend on the values of other inputs (Fig. 7.1).

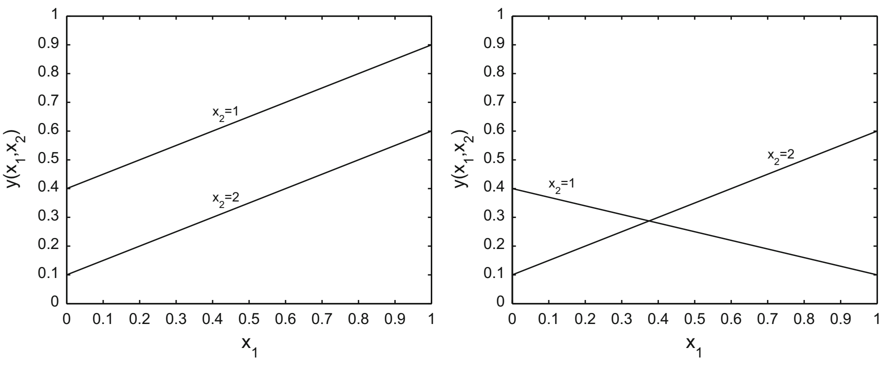


Fig. 7.1 Left panel, a function $y(x_1, x_2)$ with no $x_1 \times x_2$ interaction; right panel, a function $y(x_1, x_2)$ having $x_1 \times x_2$ interaction

The remainder of this chapter is organized as follows. Sections 7.2–7.6 emphasize methods of quantifying the *global sensitivity analysis* of a code with respect to each of its inputs and then estimating these sensitivity indices. They will also describe a companion method of visualizing the sensitivity of a code to each input based on *elementary effects*. An efficient class of designs called *one-at-a-time designs* will be introduced for estimating elementary effects. Section 7.7 describes alternative approaches and provides additional details for some of the topics presented in the earlier sections.

7.2 Classical Approaches to Sensitivity Analysis

7.2.1 Sensitivity Analysis Based on Scatterplots and Correlations

Possibly the simplest approach to sensitivity analysis uses familiar graphical and numerical tools. A scatterplot of each input versus the output of the code provides a visual assessment of the marginal effect of each input on the output. The product moment correlations between each input and the output indicate the extent to which there is *linear association* between the outputs and the input. Scatterplots are generally more informative than correlations because nonlinear relationships can be seen in plots, whereas correlations only indicate the presence of straight-line relationships.

As an example, Fig. 1.4 on page 8 plots the failure depth of pockets punched into sheet metal (the output) versus clearance and versus fillet radius, two characteristics of the machine tool used to form the pockets. The scatterplot of failure depth versus clearance shows an increasing trend, suggesting that failure depth is sensitive to clearance. However, in the scatterplot of failure depth versus fillet radius, no trend appears to be present, suggesting that failure depth may not be sensitive to fillet radius.

One limitation of marginal scatterplots is that they do not allow assessment of possible interaction effects. Three graphical methods that can be used to explore two-factor interaction effects are three-dimensional plots of the output versus pairs of inputs, two-dimensional plots that use different plotting symbols to represent the (possibly grouped) values of a second input, and a series of two-dimensional plots each of whose panels use only the data corresponding to a (possibly grouped) value of the second input. The latter are called "trellis plots." Graphical displays that allow one to investigate three-way and higher interactions are possible but typically require some form of dynamic ability to morph the figure and experience in interpretation (see, e.g., Barton (1999)).

7.2.2 Sensitivity Analysis Based on Regression Modeling

Regression analysis provides another sensitivity analysis methodology that builds on familiar tools. The method below is most effective when the design is orthogonal or nearly orthogonal and a first-order linear model in the inputs $x_1, \ldots, x_d$ (nearly) explains the majority of the variability in the output.

The regression approach to sensitivity analysis first standardizes the output $y(\boldsymbol{x})$ and all the inputs $x_1, \ldots, x_d$. If n_s runs of the simulator code have been made, each variable is standardized by subtracting that variable's mean and dividing the difference by the sample standard deviation. For example, fix an input j, $1 \leq j \leq d$, and let $x_{1,j}, \ldots, x_{n_s,j}$ denote the values of this variable for the n_s runs. Let $\overline{x}_j$ denote the mean of $x_{1,j}, \ldots, x_{n_s,j}$ and s_j their standard deviation. The standardized value of $x_{i,j}$ is defined to be

$$x^{\star}_{i,j} = \frac{x_{i,j} - \overline{x}_j}{s_j}, \quad 1 \le i \le n_s.$$

In a similar fashion, standardize the output values yielding $y^{\star}_i$, $1 \le i \le n_s$. Now fit the *first-order regression model*

$$y^{\star} = \beta^{\star}_0 + \beta^{\star}_1 x^{\star}_1 + \cdots + \beta^{\star}_d x^{\star}_d \tag{7.2.1}$$

to the standardized variables. The regression coefficients in (7.2.1) are called the *standardized regression coefficients* (SRCs); $\beta^{\star}_j$ measures the change in $y^{\star}$ due to a unit standard deviation change in input j. Because all variables have been placed on a common scale, the magnitudes of the estimated SRCs indicate the relative sensitivity of the output to each input. The output is judged most sensitive to those inputs whose SRCs are largest in absolute value.

The validity of the method depends on the overall fit of the regression model, as indicated by standard goodness-of-fit measures such as the coefficient of determination R^2. If the overall fit is poor, the SRCs do not reflect the effect of the inputs on the output. In addition, regression-based methods are most effective when the input design is orthogonal or at least space-filling so that changes in the output due to one input cannot be masked by changes in another.

Example 1.2 (Continued). Recall the data introduced in Sect. 1.2 that described the failure depth for a computational model of the operation of punching symmetric rectangular pockets in automobile steel sheets. Table 7.1 lists the regression coefficients for model (7.2.1). This analysis is likely to be reasonable because the R^2 associated with the fitted model is 0.9273. The estimated regression coefficients suggest that the output is most sensitive to *Clearance* and then, equally so, to the two inputs *Fillet Radius* and *Punch Plan View Radius*. The other inputs are of lesser importance. ♦

Input	Est. $\beta^{\star}_i$ in (7.2.1)
Clearance	0.8705
Fillet radius	0.2490
Punch plan view radius	0.2302
Width	0.0937
Length	0.0681
Lock bead distance	0.0171

Table 7.1 Estimated SRCs for the fitted standardized model (7.2.1)

Example 1.1 (Continued). Recall the data introduced in Sect. 1.2 of the computed time for a fire to reach 5 ft above a fire source. The inputs affecting this time were the room area, room height, heat loss fraction, and height of the fire source above the room. Figure 3.3 on page 82 shows the marginal relationship between the output and each input based on a 40-point Sobol′ design. From these plots, it appears that the output is most sensitive to room area and not very sensitive to the remaining inputs. We perform a sensitivity analysis of the output function based on the 40-point training data for this example. Fitted model (7.2.1) has $R^2 = 0.98$ suggesting that the output is highly linear in the four inputs and the regression approach to sensitivity analysis is likely to be accurate. Table 7.2 lists the regression coefficients for this model. These values suggest that the single most important input is *Room area*, followed by *Fire height*, *Room height*, and lastly *Heat loss fraction*. ♦

Input	Est. $\beta_i^\star$ in (7.2.1)
Heat loss frac.	0.1283
Fire height	0.5347
Room height	0.3426
Room area	0.9066

Table 7.2 Estimated SRCs for the fitted standardized model (7.2.1)

There are a number of variants on regression-based models. Partial correlation coefficients (PCCs) between the output and the inputs can be used to assess sensitivity. PCCs measure the strength of the linear relationship between the output and a given input, after adjusting for any linear effects of the other inputs. The relative sizes of PCCs are used to assess the sensitivity of the output to the inputs.

As for SRCs, the same two circumstances will compromise the validity of PCCs. If the overall fit of the model is poor, or there is a high degree of collinearity among the predictors, PCCs need not provide accurate information about the sensitivity of the output to the inputs.

A third variant of the regression approach finds rank transforms of both the inputs and the outputs. The rank transformation is carried out as follows. Suppose that a variable has N values; assign rank 1 to the lowest value, rank 2 to the next lowest, and rank N to the largest value. Use the average rank for ties. Then fit a *first-order regression model* to the transformed data. The estimated standardized regression coefficients or partial correlations are used to assess the sensitivity of the output to the inputs. Once again, if the overall fit of the first-order regression model is poor or collinearity masks the effects of one or more inputs, the use of standardized regression coefficients or partial correlations need not adequately describe the sensitivity of the output to the inputs.

In practice, it has been observed that fitted regression models based on rank transformed data often have higher R^2 values than based on standardized data. This phenomenon can occur because the rank transformation can remove (certain) nonlinearities present in the original data. Thus, when monotone (but nonlinear) trends are present, there are some advantages to conducting a sensitivity analysis using the rank transformed data. However, when one uses the rank transformed data, one must keep in mind that the resulting sensitivity measures give information on the sensitivity of the rank transformed output to the rank transformed inputs, rather than on the original variables.

A method that takes explicit account of the statistical significance of the estimated regression coefficients is a stepwise regression algorithm applied to the standardized inputs. For example, if a forward stepwise regression is used, the first variable entered would be considered the most influential input, the second variable entered would be considered the second most influential input, etc. As is usual in stepwise regression, one continues until the amount of variation explained by adding further variables is not considered meaningful according to some criterion selected by the user. Criteria such as the mean squared error, the F-statistic for testing whether the added variable significantly improves the model, Akaike's Information Criterion (the "AIC"), the coefficient of determination R^2, or the adjusted R^2 can be used to determine when to stop the stepwise regression. For more on stepwise regression, see any standard text on regression, for example, Draper and Smith (1981).

Whether one uses standardized or rank transformed data, there is no information about possible interactions or non-monotone effects of variables if only first-order models are fit. If one has reason to believe that interactions are present, or that the relation between the output and some of the inputs is nonlinear and non-monotone, these regression methods will not give reliable information about sensitivities. One may wish to consider fitting higher-order models such as a second-order response surface to the output. Such a model allows one to explore second-order (quadratic) effects of inputs and two-factor interaction (cross-product) effects. For more on response surface methods, see Box and Draper (1987).

7.3 Sensitivity Analysis Based on Elementary Effects

The elementary effects (EEs) of a function $y(\boldsymbol{x}) = y(x_1, \ldots, x_d)$ having d inputs measure the sensitivity of $y(\boldsymbol{x})$ to each x_j by directly evaluating the change in $y(\boldsymbol{x})$ when x_j alone is altered. From a geometric viewpoint, EEs are the slopes of secant lines parallel to each of the input axes. In symbols, given $j \in \{1, \ldots, d\}$, the j^{th} EE of $y(\boldsymbol{x})$ at distance Δ is

$$d_j(\boldsymbol{x}) = \frac{y(x_1, \ldots, x_{j-1}, x_j + \Delta, x_{j+1}, \ldots, x_d) - y(\boldsymbol{x})}{\Delta} .$$

Specifically, $d_j(\boldsymbol{x})$ is the slope of the secant line connecting $y(\boldsymbol{x})$ and $y(\boldsymbol{x}+\Delta\boldsymbol{e}_j)$ where $\boldsymbol{e}_j = (0, 0, \ldots, 1, 0, \ldots, 0)$ is the j^{th} unit vector. For "small" Δ, $d_j(\boldsymbol{x})$ is a numerical approximation to the j^{th} partial derivative of $y(\boldsymbol{x})$ with respect to x_j evaluated at $\boldsymbol{x}$ and hence is a local sensitivity measure. However, in most of the literature, EEs are evaluated for "large" Δ at a widely sampled set of inputs $\boldsymbol{x}$ and hence are global sensitivity measures measuring the (normalized) overall change in the output as each input moves parallel to its axis.

Example 7.1. To gain intuition about the interpretation of EEs, consider the following simple analytic "output" function:

$$y(\boldsymbol{x}) = 1.0 + 1.5x_2 + 1.5x_3 + 0.6x_4 + 1.7x_4^2 + 0.7x_5 + 0.8x_6 + 0.5x_5 \times x_6 \quad (7.3.1)$$

of $d = 6$ inputs where $\boldsymbol{x} = (x_1, x_2, x_3, x_4, x_5, x_6)$, and $x_j \in [0, 1]$ for $j = 1, \ldots, 6$. Notice that $y(\boldsymbol{x})$ is functionally independent of x_1, is linear in x_2 and x_3, is nonlinear in x_4, and contains an interaction in x_5 and x_6.

Straightforward algebra gives the value $y(\boldsymbol{x} + \Delta \boldsymbol{e}_j) - y(\boldsymbol{x})$, $j = 1, \ldots, 6$, and hence the EEs of $y(\boldsymbol{x})$ can be calculated analytically as

1. $d_1(\boldsymbol{x}) = 0$,
2. $d_2(\boldsymbol{x}) = 1.5 = d_3(\boldsymbol{x})$,
3. $d_4(\boldsymbol{x}) = +0.6 + 1.7\Delta + 3.4x_4$,
4. $d_5(\boldsymbol{x}) = +0.7 + 0.5x_6$, and $d_6(\boldsymbol{x}) = +0.8 + 0.5x_5$.

The EEs for this example are interpreted as follows. The EE of the totally inactive variable x_1 is zero because $y(\boldsymbol{x})$ is functionally independent of x_1. The EEs of the additive linear terms x_2 and x_3 are the *same* nonzero constant, 1.5, and hence (7.3.1) is judged to be equally sensitive to x_2 and x_3. The EE of the quadratic term x_4 depends on *both* the x_4 and Δ; hence for *fixed* Δ, $d_4(\boldsymbol{x})$ will vary with x_4 alone. Lastly, for the interacting x_5 and x_6 inputs, the EE $d_5(\boldsymbol{x})$ depends on x_6, while $d_6(\boldsymbol{x})$ depends on x_5.

In general, the EEs of additive linear terms are *local sensitivity measures*. The global sensitivity viewpoint assesses the sensitivity of $y(\boldsymbol{x})$ to input x_j by the *change in* $y(\boldsymbol{x})$ as x_j *moves over its range*. For example, suppose that the range of x_3 were *modified* to be $[0, 2]$. Then the larger range of x_3 compared with x_2 would mean that any reasonable assessment of the global sensitivity of $y(\boldsymbol{x})$ should conclude that x_3 is "more active" than x_2. ♦

EEs can be used as an exploratory data analysis tool, as follows. Suppose that each $d_j(\boldsymbol{x})$, $j = 1, \ldots, d$, has been computed for r vectors, say $\boldsymbol{x}_1^j, \ldots, \boldsymbol{x}_r^j$. Let $\overline{d_j}$ and S_j denote the sample mean and sample standard deviation, respectively, of $d_j(\boldsymbol{x}_1^j), \ldots, d_j(\boldsymbol{x}_r^j)$. Then as Morris (1991) states, an input x_j having

- a *small* $\overline{d_j}$ and *small* S_j is non-influential;
- a *large* $\overline{d_j}$ and *small* S_j has a strong linear effect on $y(\boldsymbol{x})$;
- a *large* S_j (and either a *large* or *small* $\overline{d_j}$) either has a nonlinear effect in x_j or x_j has strong interactions with other inputs.

Example 7.1 *(Continued).* To illustrate, consider $y(\boldsymbol{x})$ given by (7.3.1). Suppose that $y(\boldsymbol{x})$ is evaluated for each row of Table 7.3. Table 7.3 contains five *blocks* of seven rows each; the first row of every block is in boldface font. The difference between the $y(\boldsymbol{x})$ values computed from consecutive pairs of rows within each block provide one $d_j(\boldsymbol{x})$ value. Note that every such pair of rows differs by ± 0.3 in a single x_j location. In all, these output function evaluations provide $r = 5$ $d_j(\boldsymbol{x})$ evaluations for five different $\boldsymbol{x}$. The construction of the input table, "the design" of these runs,

x_1	x_2	x_3	x_4	x_5	x_6
0.50	**0.60**	**0.50**	**0.40**	**0.50**	**0.35**
0.80	0.60	0.50	0.40	0.50	0.35
0.80	0.90	0.50	0.40	0.50	0.35
0.80	0.90	0.80	0.40	0.50	0.35
0.80	0.90	0.80	0.10	0.50	0.35
0.80	0.90	0.80	0.10	0.20	0.35
0.80	0.90	0.80	0.10	0.20	0.05
0.85	**0.3**	**0.9**	**0.65**	**0.3**	**0.40**
0.55	0.30	0.90	0.65	0.3	0.40
0.55	0.00	0.90	0.65	0.30	0.40
0.55	0.00	0.60	0.65	0.30	0.40
0.55	0.00	0.60	0.95	0.30	0.40
0.55	0.00	0.60	0.95	0.60	0.40
0.55	0.00	0.60	0.95	0.60	0.10
0.65	**0.00**	**0.35**	**0.75**	**0.45**	**0.60**
0.35	0.00	0.35	0.75	0.45	0.60
0.35	0.3	0.35	0.75	0.45	0.60
0.35	0.3	0.05	0.75	0.45	0.60
0.35	0.3	0.05	0.45	0.45	0.60
0.35	0.3	0.05	0.45	0.75	0.60
0.35	0.3	0.05	0.45	0.75	0.90
0.9	**0.05**	**0.35**	**0.05**	**0.4**	**1.0**
0.60	0.05	0.35	0.05	0.40	1.00
0.60	0.35	0.35	0.05	0.40	1.00
0.60	0.35	0.05	0.05	0.40	1.00
0.60	0.35	0.05	0.35	0.40	1.00
0.60	0.35	0.05	0.35	0.10	1.00
0.60	0.35	0.05	0.35	0.10	0.70
0.40	**0.35**	**0.60**	**0.00**	**0.35**	**0.60**
0.10	0.35	0.60	0.00	0.35	0.60
0.10	0.05	0.60	0.00	0.35	0.60
0.10	0.05	0.90	0.00	0.35	0.60
0.10	0.05	0.90	0.30	0.35	0.60
0.10	0.05	0.90	0.30	0.05	0.60
0.10	0.05	0.90	0.30	0.05	0.30

Table 7.3 An OAT design with $r = 5$ complete tours for a $d = 6$ input function with domain $[0, 1]^6$ and $|\Delta| = 0.30$

will be discussed in the subsequent paragraphs. Here, only the interpretation of the plot is considered.

Figure 7.2 plots the $\overline{d_j}$ and S_j values from the individual EEs computed in the previous paragraph. Because $S_1 = 0 = S_2 = S_3$, the plot shows that $d_1(\boldsymbol{x})$, $d_2(\boldsymbol{x})$, and $d_3(\boldsymbol{x})$ are each constant and have values 0.0, 1.5, and 1.5, respectively, (as the theoretical calculations showed). Hence these $(\overline{d_j}, S_j)$ points are interpreted as saying that $y(\boldsymbol{x})$ is functionally independent of x_1 and contains an additive linear term in each of x_2 and x_3. Because $S_j > 0$ for $j = 4, 5$, and 6, the corresponding $d_j(\boldsymbol{x})$ values vary with $\boldsymbol{x}$. Hence one can conclude that either $y(\boldsymbol{x})$ is not linear in x_j or that x_j interacts with the other inputs. ♦

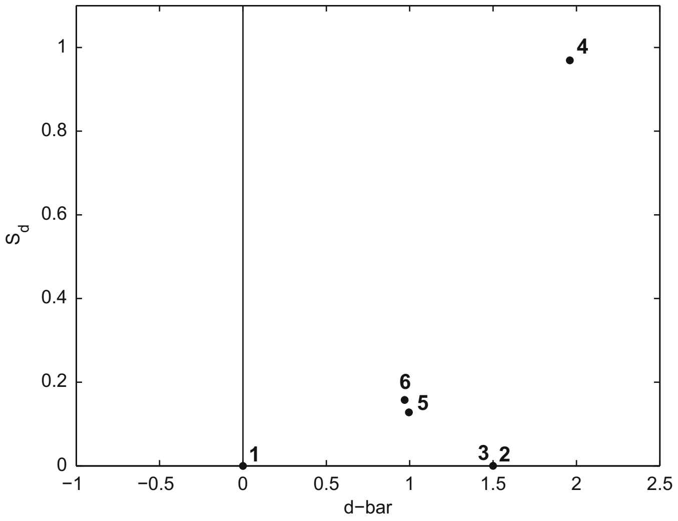


Fig. 7.2 Plot of $\overline{d_j}$ and S_j values for the EEs computed for function $y(\boldsymbol{x})$ in (7.3.1) using the design in Table 7.3

How should one select inputs in order to efficiently estimate EEs of $y(\boldsymbol{x})$? By definition, each $d_j(\boldsymbol{x})$ requires two function evaluations. Hence, at first impulse, one might think a total of $2 \times r$ function evaluations would be required to estimate r EEs. Morris (1991) proposed a more efficient *one-at-time (OAT) sampling design* for estimating the EEs when the input region is rectangular. This method is particularly useful for providing a sensitivity analysis of an *expensive* black-box computer simulator.

To introduce the method, consider evaluating an arbitrary function of five inputs:

$$y(\boldsymbol{x}) = y(x_1, x_2, x_3, x_4, x_5), \quad \boldsymbol{x} \in [0, 1]^5$$

at each of the six (input) rows listed in Table 7.4. The $y(\boldsymbol{x})$ evaluations for Runs 1 and 2 can used to compute $d_1(0.8, 0.7, 1.0, 0.7, 0.7)$ for $\Delta = -0.3$. Similarly the $y(\boldsymbol{x})$ differences for the succeeding consecutive pairs of rows provide, in order, estimates of $d_2(\boldsymbol{x})$, $d_3(\boldsymbol{x})$, $d_4(\boldsymbol{x})$, and $d_5(\boldsymbol{x})$ for different $\boldsymbol{x}$ but each using $|\Delta| = 0.3$.

Run	x_1	x_2	x_3	x_4	x_5
1	0.8	0.7	1.0	0.7	0.7
2	0.5	0.7	1.0	0.7	0.7
3	0.5	1.0	1.0	0.7	0.7
4	0.5	1.0	0.7	0.7	0.7
5	0.5	1.0	0.7	1.0	0.7
6	0.5	1.0	0.7	1.0	0.4

Table 7.4 Six input vectors at which a function of five inputs $(x_1, x_2, x_3, x_4, x_5)$ is to be evaluated

In general, a *tour* of the input space is a $(d+1)\times d$ matrix having the property that successive rows differ in a single input by a given $\pm\Delta$ and each input is varied for one pair of consecutive rows. Tours are determined by a starting $\boldsymbol{x}$, by a permutation of $(1, 2, \ldots, d)$ that specifies the input to be modified in successive pairs of rows, by the choice of a $\Delta > 0$, and by a $d \times 1$ vector $(\pm 1, \ldots, \pm 1)$ of directional movements for Δ in the successive pairs of rows. For example, the first tour in Table 7.4 uses starting vector $\boldsymbol{x} = (0.8, 0.7, 1.0, 0.7, 0.7)$, alters the inputs in succeeding rows in the order $(1, 2, 3, 4, 5)$ with $\Delta = 0.3$ and signs $(-1, +1, -1, +1, -1)$. Each row of the tour is a valid input of the function because it is an element of $[0, 1]^5$.

For a given distance $|\Delta|$, a Morris (1991) OAT design is a collection of tours with each tour starting at a randomly selected point of the input space. In his examples Morris (1991) selects the magnitude of Δ to be 30% of the common range of each scaled variable, takes a random permutation of $(1, \ldots, d)$, makes a random selection of the directional sign to be associated with each input, and selects the starting $\boldsymbol{x}$ *randomly* from a gridding of the input space; the above are restricted so that every row of the design is a valid input.

Example 1.3 (Continued). This example constructs EEs based on an OAT design for the borehole function. For convenience the formula, inputs, and input ranges of the borehole function are repeated in (7.3.2) and Table 7.5:

$$y(\boldsymbol{x}) = \frac{2\pi T_u (H_u - H_l)}{\ell n(r/r_w)\left[1 + \frac{2LT_u}{\ell n(r/r_w) r_w^2 K_w} + \frac{T_u}{T_l}\right]} . \tag{7.3.2}$$

Notation	Input	Units	Range
Radius of influence	r	m	$[100, 50000]$
Radius of the borehole	r_w	m	$[0.05, 0.15]$
Transmissivity of the upper aquifer	T_u	m^2/year	$[63070, 115600]$
Potentiometric head of the upper aquifer	H_u	m	$[990, 1110]$
Transmissivity of the lower aquifer	T_l	m^2/year	$[63.1, 116]$
Potentiometric head of the lower aquifer	H_l	m	$[700, 820]$
Length of the borehole	L	m	$[1120, 1680]$
Hydraulic conductivity of borehole	K_w	m/year	$[9855, 12045]$

Table 7.5 Inputs and ranges of the borehole function from (7.3.2)

A Morris OAT design with $r = 5$ tours based on random starting points was constructed where Δ was selected to be approximately 40% of the range of each input. The OAT design is listed in Table 7.6 with the starting points in bold font.

The $\overline{d_j}$ and S_j values computed from this design, $j = 1, \ldots, 8$, are listed in Table 7.7. A plot of the $(\overline{d_j}, S_j)$ is shown in Fig. 7.3 using the labels in Table 7.7. From the signs and magnitudes of the EEs, it is seen that the annual flow rate increases greatly in r_w, increases at a slower rate in H_u, and decreases in H_l and L. None of r, T_u, T_l, or K_w appear to be important determinants of the annual flow rate. The large S_2 suggests that the effect of r_w is either nonlinear or has strong interactions with other inputs. ♦

r	r_w	T_u	H_u	T_l	H_l	L	K_w
14483	**0.098**	**103714**	**109.37**	**1032.4**	**764.2**	**1660.8**	**9899.2**
14483	0.098	82702	109.37	1032.4	764.2	1660.8	9899.2
14483	0.098	82702	88.214	1032.4	764.2	1660.8	9899.2
14483	0.098	82702	88.214	1032.4	812.2	1660.8	9899.2
14483	0.098	82702	88.214	1080.4	812.2	1660.8	9899.2
14483	0.098	82702	88.214	1080.4	812.2	1436.8	9899.2
34123	0.098	82702	88.214	1080.4	812.2	1436.8	9899.2
34123	0.098	82702	88.214	1080.4	812.2	1436.8	10775
34123	0.058	82702	88.214	1080.4	812.2	1436.8	10775
2083.8	**0.144**	**87266**	**112.05**	**1004.5**	**755.3**	**1419.8**	**11771**
2083.8	0.104	87266	112.05	1004.5	755.3	1419.8	11771
2083.8	0.104	66254	112.05	1004.5	755.3	1419.8	11771
2083.8	0.104	66254	90.886	1004.5	755.3	1419.8	11771
2083.8	0.104	66254	90.886	1004.5	755.3	1643.8	11771
2083.8	0.104	66254	90.886	1004.5	707.3	1643.8	11771
2083.8	0.104	66254	90.886	1052.5	707.3	1643.8	11771
2083.8	0.104	66254	90.886	1052.5	707.3	1643.8	10895
21724	0.104	66254	90.886	1052.5	707.3	1643.8	10895
12003	**0.116**	**105837**	**100.82**	**1041.6**	**747.3**	**1329.3**	**11638**
12003	0.116	105837	100.82	1041.6	747.3	1329.3	10762
12003	0.076	105837	100.82	1041.6	747.3	1329.3	10762
12003	0.076	105837	100.82	1041.6	747.3	1553.3	10762
12003	0.076	105837	100.82	993.64	747.3	1553.3	10762
12003	0.076	105837	100.82	993.64	795.3	1553.3	10762
31643	0.076	105837	100.82	993.64	795.3	1553.3	10762
31643	0.076	105837	79.665	993.64	795.3	1553.3	10762
31643	0.076	84825	79.665	993.64	795.3	1553.3	10762
28171	**0.110**	**81111**	**91.96**	**1060.3**	**735.2**	**1148.3**	**10275**
8531.3	0.110	81111	91.96	1060.3	735.2	1148.3	10275
8531.3	0.110	81111	113.11	1060.3	735.2	1148.3	10275
8531.3	0.110	81111	113.11	1060.3	735.2	1372.3	10275
8531.3	0.110	81111	113.11	1108.3	735.2	1372.3	10275
8531.3	0.110	81111	113.11	1108.3	783.2	1372.3	10275
8531.3	0.110	81111	113.11	1108.3	783.2	1372.3	11151
8531.3	0.110	102123	113.11	1108.3	783.2	1372.3	11151
8531.3	0.070	102123	113.11	1108.3	783.2	1372.3	11151
35115	**0.098**	**90980**	**64.703**	**1055.5**	**726.7**	**1238.8**	**11527**
35115	0.098	69968	64.703	1055.5	726.7	1238.8	11527
35115	0.098	69968	64.703	1103.5	726.7	1238.8	11527
35115	0.058	69968	64.703	1103.5	726.7	1238.8	11527
35115	0.058	69968	64.703	1103.5	726.7	1238.8	10651
15475	0.058	69968	64.703	1103.5	726.7	1238.8	10651
15475	0.058	69968	64.703	1103.5	774.7	1238.8	10651
15475	0.058	69968	85.863	1103.5	774.7	1238.8	10651
15475	0.058	69968	85.863	1103.5	774.7	1462.8	10651

Table 7.6 A Morris OAT design for the borehole function (7.3.2) with $r = 5$ tours and $d = 8$ inputs. The boldface rows denote the start of a tour consisting of nine runs

A number of enhancements have been proposed to the basic Morris (1991) OAT design. Campolongo et al. (2007) suggest ways of making OAT designs more space-

	Input	$\overline{d_j}$	S_j
1	r	−1.80e−06	1.81e−06
2	$\boldsymbol{r_w}$	**1436.5**	**275.25**
3	T_u	4.55e−09	3.15e−09
4	T_l	2.58e−03	2.3257e−03
5	$\boldsymbol{H_u}$	**0.222**	**0.068**
6	$\boldsymbol{H_l}$	**−0.184**	**0.079**
7	$\boldsymbol{L}$	**−0.041**	**0.023**
8	K_w	6.81e−03	2.5132e−03

Table 7.7 The $\overline{d_j}$ and S_j values for the borehole function based on the OAT design in Table 7.6

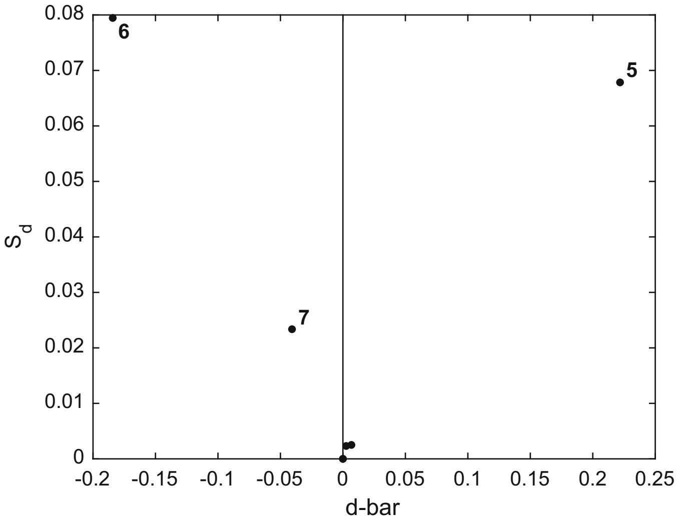


Fig. 7.3 Plot of $(\overline{d_j}, S_j)$ from Table 7.7 for the seven smallest $\overline{d_j}$ values

filling. They propose selecting the desired number, r, of tours to maximize a heuristic distance criterion between all pairs of tours. Their R package `sensitivity` implements this criterion; it was used to construct the design in Table 7.3. Pujol (2009) proposed a method of constructing OAT designs whose projections onto subsets of the input space are not collapsing. This is important if $y(\boldsymbol{x})$ depends only on a subset of "active" inputs. For example, suppose that $y(\boldsymbol{x})$ depends (primarily) on x_j where $j \in \{1, 3, 5\}$ and other x_ℓ are "inactive." If multiple $\boldsymbol{x}$ inputs from the selected EE design have common x_j for $j \in \{1, 3, 5\}$, then $y(\boldsymbol{x})$ evaluations at these $\boldsymbol{x}$ would produce (essentially) the same output and hence little information about the input–output relationship. Campolongo et al. (2011) introduced an OAT design that spreads starting points using a Sobol´ sequence and differential Δ for each pair of input vectors. Finally Sun et al. (2013) introduced an OAT design that can be used for *non-rectangular input regions* and is an alternative method to Campolongo et al. (2007) for spreading the secant lines of the design over the input space.

7.4 Global Sensitivity Analysis

Often one of the first tasks in the analysis of simulator output $y(\boldsymbol{x})$ is the rough assessment of the sensitivity of $y(\boldsymbol{x})$ to each input x_k. In a combined physical system/simulator setting, the analogous issue is the determination of the sensitivity of the *mean response of the physical system* to each input.

Sobol´ (1990, 1993) and Welch et al. (1992) described plotting methods to make such an assessment. They introduced the use of main effect plots and joint effect plots. They also define various numerical "sensitivity indices" (SIs) to make such assessments. This section will define these effect plots. It will also describe a functional ANOVA decomposition of the output $y(\boldsymbol{x})$ that is used to define "global SIs."

More formal variable screening methods have been developed for computer simulators by Linkletter et al. (2006) and Moon et al. (2012). The methodology in both papers assumes training data are available to construct a Gaussian process (GP) emulator of the output at arbitrary input sites (with Gaussian correlation function). Linkletter et al. (2006) use the (posterior distribution of the) estimated process correlation for each input to assess its impact on $y(\boldsymbol{x})$, while Moon et al. (2012) calculate each input's "total effect" index for the same purpose.

To ease the notational burden, from this point on *assume* that $y(\boldsymbol{x})$ has a *hyper-rectangular* input domain *which is taken to be* $[0, 1]^d$. If the input domain of $y(\boldsymbol{x})$ is $\prod_{j=1}^{d}[a_j, b_j]$, one should apply the methods below to the function

$$y^{\star}(x_1, \ldots, x_d) = y\,(a_1 + x_1(b_1 - a_1), \ldots, a_d + x_d(b_d - a_d)) \ .$$

When the input domain is *not* a hyper-rectangle, there are several papers that consider analogs of the effect function definitions and sensitivity indices defined below; however this topic is an area of active research (e.g., Loeppky et al. (2013)).

Section 7.4.1 introduces (uncentered) *main effect* and *joint effect* functions for a given $y(\boldsymbol{x})$, which are weighted $y(\boldsymbol{x})$ averages. Then Sect. 7.4.2 describes an ANOVA-like expansion of $y(\boldsymbol{x})$ in terms of centered and orthogonalized versions of the main and joint effect functions. Section 7.4.3 defines sensitivity indices for individual inputs and groups of inputs in terms of the variability of these functional ANOVA components. Section 7.5 provides methods for estimating these plots and indices based on a set of $y(\boldsymbol{x})$ runs. The emphasis in Sect. 7.5 will be on methods for simulators having "expensive" code runs so that limited amounts of training data are available.

7.4.1 Main Effect and Joint Effect Functions

Given a weight function of the form $w(\boldsymbol{x}) = \prod_{k=1}^{d} g_k(x_k)$ where $g_k(\cdot)$ is a probability density function on $[0, 1]$, denote the *overall mean* of $y(\cdot)$ with respect to $w(\boldsymbol{x})$ by

$$y_0 \equiv \int_0^1 \cdots \int_0^1 y(x_1, \ldots, x_d) \prod_{k=1}^{d} g_k(x_k)\, dx_k \, .$$

Thus y_0 can be interpreted as the expectation $E[y(\boldsymbol{X})]$ where $\boldsymbol{X} = (X_1, \ldots, X_d)$ has joint probability density $\prod_{k=1}^{d} g_k(x_k)$. The weighted mean is useful, for example, in scientific settings where the output depends on inputs that are not known exactly but whose uncertainty can be specified by independent input distributions. If the uncertainty in $\boldsymbol{X}$ can be specified by a weight function $w(\boldsymbol{x})$ that has nonindependent input components, then analogs of some of the results below still hold; settings where the results hold for general $w(\boldsymbol{x})$ will be described in this section. For notational simplicity, the development below assumes that the weight function corresponds to independent and identically distributed $U(0, 1)$ components so that

$$y_0 \equiv \int_0^1 \cdots \int_0^1 y(x_1, \ldots, x_d) \prod_{k=1}^{d} dx_k \, . \tag{7.4.1}$$

Similarly, the k^{th} *main effect function* of $y(\boldsymbol{x})$, $k = 1, \ldots, d$, is defined to be

$$u_k(x_k) = \int_0^1 \cdots \int_0^1 y(x_1, \ldots, x_d) \prod_{\ell \neq k} dx_\ell = E\left[y(\boldsymbol{X}) \mid X_k = x_k\right], \tag{7.4.2}$$

which is the average $y(\boldsymbol{x})$ value for fixed x_k. The expectation notation uses the fact that the components of $\boldsymbol{X}$ are independent.

The idea of averaging $y(\boldsymbol{x})$ when a single input is fixed can be extended to fixing multiple inputs. Select a non-empty subset Q of $\{1, \ldots d\}$ for which the complement $\{1, \ldots d\} \setminus Q$ is also non-empty (so that the integral (7.4.3) averages over at least one input variable). Let $\boldsymbol{x}_Q$ denote the vector of x_k with $k \in Q$, arranged in order of increasing k. Define the *joint effect function* of $y(x_1, \ldots, x_d)$ when $\boldsymbol{x}_Q$ is *fixed* to be

$$u_Q(\boldsymbol{x}_Q) = \int_0^1 \cdots \int_0^1 y(x_1, \ldots, x_d) \prod_{\ell \notin Q} dx_\ell = E\left[y(\boldsymbol{X}) \mid \boldsymbol{X}_Q = \boldsymbol{x}_Q\right], \tag{7.4.3}$$

which is the average $y(x_1, \ldots, x_d)$ value when the components of $\boldsymbol{x}_Q$ are held constant. For completeness, set

$$u_{1,2,\ldots,d}(x_1, \ldots, x_d) \equiv y(x_1, \ldots, x_d)$$

(when $Q = \{1, \ldots d\}$).

For any $Q \subset \{1, \ldots, d\}$, it is straightforward to see that the mean of the joint effect function $u_Q(\boldsymbol{x}_Q)$ with respect to *all* arguments $\boldsymbol{X}_Q$ is y_0, i.e.,

$$E\left[u_Q(\boldsymbol{X}_Q)\right] = \int_0^1 \cdots \int_0^1 u_Q(\boldsymbol{x}_Q)\, d\boldsymbol{x}_Q = y_0 \, . \tag{7.4.4}$$

The mean of $u_Q(\boldsymbol{x}_Q)$ with respect to any collection of x_k whose indices k form a *proper* subset of Q will vary and, in particular, need not be zero. Hence the $u_Q(\boldsymbol{x}_Q)$ are sometimes called *uncentered* effect functions. In contrast, the ANOVA-type centered versions of the $u_Q(\boldsymbol{x}_Q)$ that are defined in the next subsection have zero mean with respect to any set of x_k, with $k \in Q$.

Effect functions $u_Q(\boldsymbol{x}_Q)$ can be defined for nonindependent weight functions by the conditional expectation operation in (7.4.3). However, their interpretation and usefulness are more limited than in the independence case.

Example 7.2. Suppose $y(x_1, x_2) = 2x_1 + x_2$ is defined on $[0, 1]^2$. Then the overall mean and u_Q effect functions are

$$\begin{aligned} y_0 &= \int_0^1 \int_0^1 (2x_1 + x_2)\, dx_2\, dx_1 = 1.5, \\ u_1(x_1) &= \int_0^1 (2x_1 + x_2)\, dx_2 = 0.5 + 2x_1, \\ u_2(x_2) &= \int_0^1 (2x_1 + x_2)\, dx_1 = 1.0 + x_2, \text{ and} \\ u_{12}(x_1, x_2) &= y(x_1, x_2) = 2x_1 + x_2. \end{aligned}$$

Illustrating the fact (7.4.4) that y_0 is the mean of every $u_Q(\boldsymbol{X}_Q)$ with respect to $\boldsymbol{X}_Q$, it is simple to calculate that

$$\int_0^1 u_1(x_1)\, dx_1 = \int_0^1 u_2(x_2)\, dx_2 = \int_0^1 \int_0^1 u_{12}(x_1, x_2)\, dx_1\, dx_2 = 1.5.$$

As shown in Fig. 7.4, plots of the main effect functions provide accurate information of how this simple function behaves in x_1 and x_2. ♦

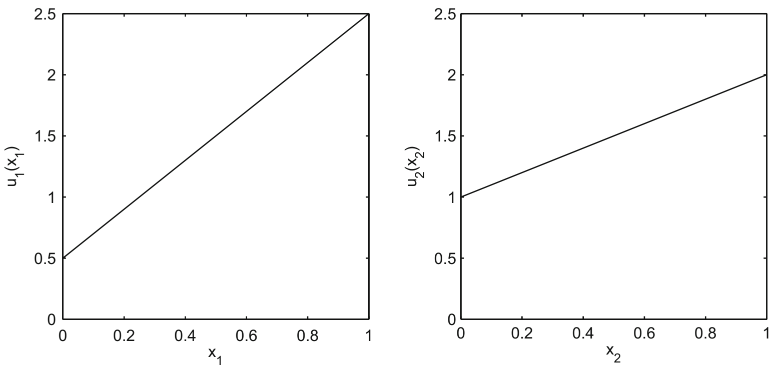


Fig. 7.4 Plots of the main effect functions $u_1(x_1)$ and $u_2(x_2)$ for $y(x_1, x_2) = 2x_1 + x_2$

Example 7.3. The so-called g-function with d inputs is defined to be

$$y(x_1, \ldots, x_d) = \prod_{k=1}^{d} \frac{|4x_k - 2| + c_k}{1 + c_k}, \tag{7.4.5}$$

where $\boldsymbol{x} \in [0,1]^d$ and $\boldsymbol{c} = (c_1, \ldots, c_d)$ has nonnegative components (Saltelli and Sobol´ (1995)).

Note that $y(\boldsymbol{x})$ is a product of functions of each input and does not involve stand-alone "linear" terms in any inputs. As the definition of the effect function states, and this example is meant to emphasize, $u_i(x_i)$ contains the contributions of every x_i component no matter whether they appear as a stand-alone term or as part of an interaction.

For fixed $c \geq 0$, the value of

$$q(x) = \frac{|4x - 2| + c}{1 + c}$$

over $x \in [0,1]$ is a pair of line segments, one over $[0, 1/2]$ and the second over $[1/2, 1]$, defined by $q(1/2) = c/(1 + c)$ and $q(0) = (2 + c)/(1 + c) = q(1)$. The function $q(x)$ is symmetric about $x = 1/2$. It is straightforward to determine that

$$\int_0^1 \frac{|4x - 2| + c}{1 + c}\, dx = 1.0\,, \tag{7.4.6}$$

for every $c \geq 0$ because this integral is the sum of the areas of two identical trapezoids (triangles when $c = 0$). The value of c determines how "active" x is in $q(x)$. Figure 7.5 shows this effect by plotting three $q(x)$ with $c \in \{0, 5, 25\}$. The variable x is more active for $q(x)$ having smaller c.

Returning to the g-function with arbitrary numbers of inputs, (7.4.5), and arbitrary vector of parameters $\boldsymbol{c} = (c_1, \ldots, c_d) \geq 0$, the main and joint effect functions of $y(\boldsymbol{x})$ are simple to calculate using (7.4.6). The overall mean is

$$y_0 = \int_0^1 \cdots \int_0^1 y(x_1, \ldots, x_d) \prod_{k=1}^{d} dx_k = \prod_{k=1}^{d} \int_0^1 \frac{|4x_k - 2| + c_k}{1 + c_k}\, dx_k = 1.0\,.$$

The k^{th} main effect function is

$$u_k(x_k) = \frac{|4x_k - 2| + c_k}{1 + c_k} \times 1 = \frac{|4x_k - 2| + c_k}{1 + c_k}\,, \quad k = 1, \ldots, d\,.$$

Thus the main effect plots of individual inputs have essentially the symmetric form shown in Fig. 7.5.

In a similar way, given a non-empty subset Q of $\{1, \ldots d\}$,

$$u_Q(\boldsymbol{x}_Q) = \prod_{k \in Q} \frac{|4x_k - 2| + c_k}{1 + c_k}\,. \tag{7.4.7}$$

For example,

$$u_{12}(x_1, x_2) = \frac{|4x_1 - 2| + c_1}{1 + c_1} \times \frac{|4x_2 - 2| + c_2}{1 + c_2}\,, \tag{7.4.8}$$

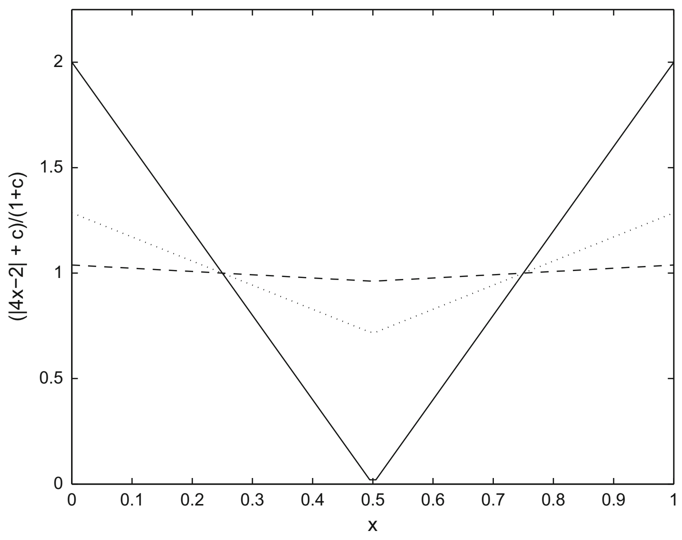


Fig. 7.5 The function $\frac{|4x-2|+c}{1+c}$ for $c = 0$ (solid line), $c = 5$ (dotted line), and $c = 25$ (dashed line)

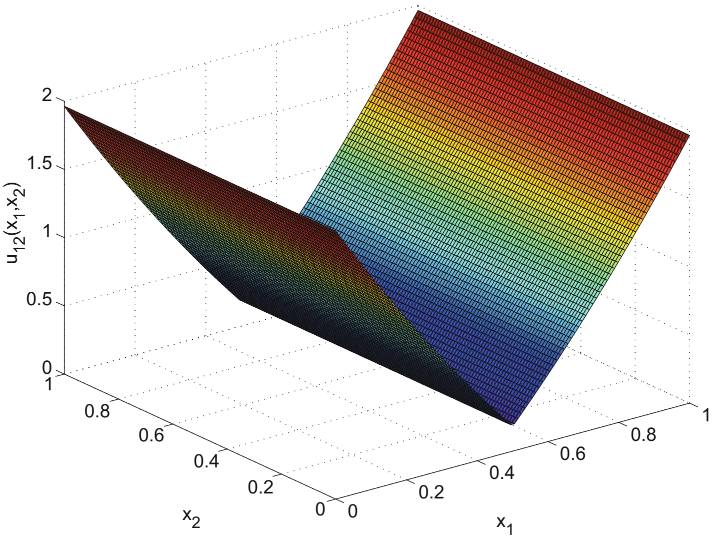


Fig. 7.6 Joint effect function (7.4.8) for $c_1 = 0.25$ and $c_2 = 10.0$

which is plotted in Fig. 7.6 for $c_1 = 0.25$ and $c_2 = 10.0$. Clearly this function shows that x_1, the input associated with the smaller c_i, is more active than x_2. Visualizations of higher-order effect functions, while more difficult to display effectively, can also be made. ♦

In sum, plots of main effect functions $(x_i, u_i(x_i))$ and joint effect functions $\big((x_i, x_j), u_{ij}(x_i, x_j)\big)$ can be used to provide a rough visual understanding of the change in the averaged $y(\boldsymbol{x})$ with respect to each single input or pairs of inputs. Section 7.5 will describe methods of estimating the $u_Q(\boldsymbol{x}_Q)$ based a set of training data obtained from the output function.

7.4.2 A Functional ANOVA Decomposition

The uncentered $u_Q(\boldsymbol{x}_Q)$ describe average $y(\boldsymbol{x})$ values; $u_Q(\boldsymbol{x}_Q)$ values are on the same scale and in the same range as $y(\boldsymbol{x})$. The sensitivity indices that we shall define shortly, assess the variability of (a centered version) of the $u_Q(\boldsymbol{x}_Q)$. Viewed with this objective in mind, the (uncentered) joint effect functions have an important defect that limits their usefulness for constructing sensitivity indices. Namely, when viewed as functions of random $\boldsymbol{X}_i$ inputs, different effect functions are, in general, correlated. For example, if X_1 and X_2 are independent $U(0, 1)$ random variables, $Cov[u_1(X_1), u_2(X_2)]$ need not equal 0.

Thus Sobol´ (1990, 1993) advocated the use of a functional ANOVA-like decomposition of $y(\boldsymbol{x})$ that modifies the uncentered joint effect functions to produce *uncorrelated* and *zero mean* versions of the $u_Q(\boldsymbol{x}_Q)$. These centered effect functions will be used to define sensitivity indices. Specifically, Sobol´ (1993) advocated use of the $y(\boldsymbol{x})$ decomposition:

$$y(\boldsymbol{x}) = y_0 + \sum_{k=1}^{d} y_k(x_k) \;+ \sum_{1\le k<j\le d} y_{kj}(x_k, x_j) + \cdots + y_{1,2,\ldots,d}(x_1, \ldots, x_d) \qquad (7.4.9)$$

where $x_1, \ldots, x_d$ are the components of $\boldsymbol{x}$ and the terms of (7.4.9) are defined as follows. For any fixed k, $k = 1, \ldots, d$, define

$$y_k(x_k) = u_k(x_k) - y_0 = \int_0^1 \cdots \int_0^1 y(\boldsymbol{x}) \prod_{\ell\neq k} dx_\ell \; - y_0 \qquad (7.4.10)$$

to be the centered main effect function of input x_k. For any fixed (k, j), $1 \le k < j \le d$, define

$$\begin{aligned} y_{kj}(x_k, x_j) &= u_{kj}(x_k, x_j) - y_k(x_k) - y_j(x_j) - y_0 \\ &= \int_0^1 \cdots \int_0^1 y(\boldsymbol{x}) \prod_{\ell\neq k,j} dx_\ell \; - y_k(x_k) - y_j(x_j) - y_0 \end{aligned} \qquad (7.4.11)$$

to be the centered interaction effect $u_{kj}(x_k, x_j)$. Higher-order interaction terms are defined in a recursive manner; if Q is a non-empty subset of $\{1, \ldots d\}$,

$$y_Q(\boldsymbol{x}_Q) = u_Q(\boldsymbol{x}_Q) - \sum_E y_E(\boldsymbol{x}_E) - y_0\,, \tag{7.4.12}$$

where the sum is over all *non-empty proper* subsets E of Q; $E \subset Q$ is proper provided $E \neq Q$. For example, if $y(\boldsymbol{x})$ has three or more inputs,

$$\begin{aligned} y_{123}(x_1, x_2, x_3) = u_{123}(x_1, x_2, x_3) &- y_{12}(x_1, x_2) - y_{13}(x_1, x_3) - y_{23}(x_2, x_3) \\ &- y_1(x_1) - y_2(x_2) - y_3(x_3) - y_0\,. \end{aligned}$$

In particular,

$$\begin{aligned} y_{1,2,\ldots,d}(x_1, x_2, \ldots, x_d) &= u_{1,2,\ldots,d}(x_1, x_2, \ldots, x_d) - \sum_E y_E(\boldsymbol{x}_E) - y_0 \\ &= y(x_1, x_2, \ldots, x_d) - \sum_E y_E(\boldsymbol{x}_E) - y_0\,, \end{aligned}$$

where the sum is over all non-empty proper subsets E of $\{1, \ldots, d\}$. This final equation is the decomposition (7.4.9). Notice that even if the effect functions are $y(\boldsymbol{x})$ averages defined in terms of an arbitrary weight function, (7.4.9) still holds by construction (see Hoeffding (1948) who implicitly uses this expansion or Van Der Vaart (1998) who derives it from an alternate point of view).

However, if the weight function is defined by mutually independent input component distributions, Sect. 7.7 shows that the centered effect functions have two properties that make them extremely useful for defining sensitivity indices. First, each $y_Q(\boldsymbol{x}_Q)$, $Q \subset \{1, \ldots, d\}$, has *zero mean* when averaged over any collection of x_i whose indices i are a subset of Q. Suppose that $Q = \{i_1, \ldots, i_s\}$ and $i_k \in Q$, then

$$E\left[y_{i_1,\ldots,i_s}(x_{i_1}, \ldots, X_{i_k}, \ldots, x_{i_s})\right] = \int_0^1 y_{i_1,\ldots,i_s}(x_{i_1}, \ldots, x_{i_s})\; dx_{i_k} = 0\,, \tag{7.4.13}$$

for any fixed values x_{i_ℓ}, $i_\ell \in Q \setminus \{i_k\}$. The second property of the $y_Q(\boldsymbol{x}_Q)$ is that they are *orthogonal* meaning that for any $(i_1, \ldots, i_s) \neq (j_1, \ldots, j_t)$,

$$\begin{aligned} &Cov\left[y_{i_1,\ldots,i_s}(X_{i_1}, \ldots, X_{i_s}), y_{j_1,\ldots,j_t}(X_{j_1}, \ldots, X_{j_t})\right] = \\ &\int_0^1 \cdots \int_0^1 y_{i_1,\ldots,i_s}(x_{i_1}, \ldots, x_{i_s}) \times y_{j_1,\ldots,j_t}(x_{j_1}, \ldots, x_{j_t}) \prod_\ell dx_\ell = 0\,, \end{aligned} \tag{7.4.14}$$

where the product in (7.4.14) is over all $\ell \in \{i_1, \ldots, i_s\} \cup \{j_1, \ldots, j_t\}$.

Example 7.2 (Continued). Using y_0, and the $u_Q(\cdot)$ effect functions calculated previously, algebra gives

$$y_1(x_1) = u_1(x_1) - y_0 = 0.5 + 2x_1 - 1.5 = -1 + 2x_1\,,$$

$$y_2(x_2) = u_2(x_2) - y_0 = 1.0 + x_2 - 1.5 = -0.5 + x_2\,, \text{ and}$$
$$y_{12}(x_1, x_2) = u_{12}(x_1, x_2) - y_1(x_1) - y_2(x_2) - y_0 = 0\,.$$

The function $y_{12}(x_1, x_2) = 0$ states that there is no interaction between x_1 and x_2. Calculus allows one to verify the properties (7.4.13) and (7.4.14) for this example because

$$E\left[y_1(X_1)\right] = \int_0^1 (-1 + 2x_1)\, dx_1 = 0\,,$$

$$E\left[y_2(X_2)\right] = \int_0^1 (-0.5 + x_2)\, dx_2 = 0\,,$$

$$E\left[y_{12}(X_1, x_2)\right] = 0 = E\left[y_{12}(x_1, X_2)\right]\,,$$

as well as, for example, $Cov[y_1(X_1), y_{12}(X_1, X_2)] = 0$. ♦

Example 7.3 *(Continued).* Recalling formula (7.4.7), the centered effect function

$$y_k(x_k) = u_k(x_k) - y_0 = \frac{|4x_k - 2| + c_k}{1 + c_k} - 1 = \frac{|4x_k - 2| - 1}{1 + c_k},$$

for $1 \le k \le d$, while

$$\begin{aligned} y_{kj}(x_k, x_j) &= u_{kj}(x_k, x_j) - y_k(x_k) - y_j(x_j) - y_0 \\ &= \frac{|4x_k - 2| + c_k}{1 + c_k} \times \frac{|4x_j - 2| + c_j}{1 + c_j} - \frac{|4x_k - 2| - 1}{1 + c_k} - \frac{|4x_j - 2| - 1}{1 + c_j} - 1, \end{aligned}$$

for $1 \le k < j \le d$. ♦

When $X_1, \ldots, X_d$ are mutually independent, the corrected effects can be used to partition the variance of $y(\boldsymbol{X})$ into components of variance that define global sensitivity indices. As always we take $X_1, \ldots, X_d$ to have independent $U(0, 1)$ distributions. Recalling that

$$y_0 = E\left[y(\boldsymbol{X})\right],$$

the *total variance*, V, of $y(\boldsymbol{X})$ is

$$V \equiv E\left[(y(\boldsymbol{X}) - y_0)^2\right] = E\left[y^2(\boldsymbol{X})\right] - y_0^2\,.$$

Recalling that for any subset $Q \subset \{1, \ldots, d\}$, $y_Q(\boldsymbol{X}_Q)$ has mean zero,

$$V_Q \equiv Var\left[y_Q(\boldsymbol{X}_Q)\right] = E\left[y_Q^2(\boldsymbol{X}_Q)\right]$$

denotes the variance of the term $y_Q(\boldsymbol{X}_Q)$ in (7.4.9). Using (7.4.9) and the fact that the covariances of different $y_Q(\boldsymbol{X}_Q)$ terms are zero, we calculate

$$V = E\left[(y(\boldsymbol{X}) - y_0)^2\right]$$

$$= E\left[\left(\sum_{k=1}^{d} y_k(X_k) + \sum_{k<j} y_{kj}(X_k, X_j) + \cdots + y_{1,2,\ldots,d}(X_1, \ldots, X_d)\right)^2\right]$$

$$= \sum_{k=1}^{d} E\left[y_k^2(X_k)\right] + \sum_{k<j} E\left[y_{kj}^2(X_k, X_j)\right] + \cdots + E\left[y_{1,2,\ldots,d}^2(X_1, \ldots, X_d)\right]$$

$$+ \sum E\left[y_E(\boldsymbol{X}_E)\, y_{E^\star}(\boldsymbol{X}_{E^\star})\right], \tag{7.4.15}$$

where the sum in (7.4.15) is over all non-empty subsets E and $E^\star$ of $\{1, \ldots, d\}$ for which $E \neq E^\star$. Thus

$$V = \sum_{k=1}^{d} E\left[y_k^2(X_k)\right] + \sum_{k<j} E\left[y_{kj}^2(X_k, X_j)\right] + \cdots$$

$$+ E\left[y_{1,2,\ldots,d}^2(X_1, \ldots, X_d)\right]$$

$$= \sum_{k=1}^{d} V_k + \sum_{k<j} V_{kj} + \cdots + V_{1,2,\ldots,d}\,. \tag{7.4.16}$$

It should be reiterated that the sum (7.4.16) requires that the covariances of different $y_Q(\boldsymbol{X}_Q)$ terms be zero which was a consequence of the independence of the individual input distributions.

The functional decomposition (7.4.9) can, in a more formal way, be modified to result in the classical ANOVA decomposition of a model with d quantitative factors. Suppose that, instead of $\boldsymbol{X}$ taking a uniform distribution over $[0, 1]^d$, the input factor space is regarded as the *discrete* set of points $\{0, \frac{1}{n-1}, \ldots, \frac{n-2}{n-1}, 1\}^d$ with a *discrete uniform distribution* over these n values. This would arise, for example, if the inputs formed an n^d factorial with n levels for each factor that are coded $0, \frac{1}{n-1}, \ldots, \frac{n-2}{n-1}$, 1. Replacing each *integral* over $[0, 1]$ that defines a term in (7.4.9) by an *average* over the n discrete values, y_0 becomes $\overline{y}$, the overall mean of all the $y(\cdot)$, the $\{y_i\}_i$ become the usual ANOVA estimates of main effects in a complete factorial, the $\{y_{ij}\}_{ij}$ become the usual ANOVA estimates of two-factor interactions in a complete factorial, and so on. Finally, it is clear that V in (7.4.16) is the mean corrected sum of squares of all the $y(\cdot)$, V_i is the sum of squares for the i^{th} factor, and so forth. Thus, the decomposition (7.4.16) is the usual ANOVA decomposition into sums of squares for main effects and higher-way interactions.

7.4.3 Global Sensitivity Indices

For any subset $Q \subset \{1, \ldots, d\}$, define the sensitivity index (SI) of $y(\boldsymbol{x})$ with respect to the set of inputs x_k, $k \in Q$, to be

$$S_Q = \frac{V_Q}{V}.$$

In particular, the sensitivity index corresponding to $Q = \{k\}$ is called the *first-order or main effect sensitivity index* and is denoted by S_k; intuitively, S_k measures the proportion of the variation V that is due to input x_k. The sensitivity index corresponding to $Q = \{k, j\}$, $1 \leq k < j \leq d$, is called the *two-way sensitivity index* of the k^{th} and j^{th} inputs; it is denoted by S_{kj}. S_{kj} measures the proportion of V that is due to the joint effects of the inputs x_k and x_j. Higher-order sensitivity indices, simply called joint effect sensitivity indices, are defined analogously. From (7.4.16) the sensitivity indices satisfy

$$\sum_{k=1}^{d} S_k + \sum_{1 \leq k < j \leq d} S_{kj} + \cdots + S_{1,2,\ldots,d} = 1\,.$$

We illustrate these definitions and the interpretations of SIs with examples.

Example 7.2 (Continued). For $y(x_1, x_2) = 2x_1 + x_2$, recall that $y_1(x_1)$, $y_2(x_2)$, and $y_{12}(x_1, x_2)$ calculated previously, give

$$\begin{aligned}
V &= Var\left[y(X_1, X_2)\right] = Var\left[2X_1 + X_2\right] = 4/12 + 1/12 = 5/12\,,\\
V_1 &= Var\left[y_1(X_1)\right] = Var\left[-1 + 2X_1\right] = 4/12\,,\\
V_2 &= Var\left[y_2(X_2)\right] = Var\left[-0.5 + X_2\right] = 1/12\,, \text{ and}\\
V_{12} &= Var\left[y_{12}(X_1, X_2)\right] = Var\left[0\right] = 0\,,
\end{aligned}$$

so that $V = V_1 + V_2 + V_{12}$ and

$$S_1 = \frac{4/12}{5/12} = 0.8,\ S_2 = \frac{1/12}{5/12} = 0.2,\ \text{and}\ S_{12} = 0.0.$$

The interpretation of these values coincides with our intuition about $y(x_1, x_2)$: x_1 is more important than x_2, while there is no interaction between x_1 and x_2. The only deviation from our intuition is that, based on the functional relationship, the reader might have assessed that x_1 was *twice* as important as x_2, whereas the variance computations used by global sensitivity indices rely on the fact that $Var[2X_1] = 4Var[X_1]$. ♦

Before considering additional examples, we use the S_Q sensitivity indices to define the so-called *total sensitivity index* (TSI) of $y(\boldsymbol{x})$ with respect to a given input x_k, $1 \leq k \leq d$; T_k is meant to include interactions of x_k with all other inputs. The *total sensitivity of input k* is defined to be *sum* of all the sensitivity indices involving the k^{th} input; in symbols,

$$T_k = S_k + \sum_{j<k} S_{jk} + \sum_{j>k} S_{kj} + \cdots + S_{1,2,\ldots,d}\,. \tag{7.4.17}$$

For this example, when $d = 3$,

$$T_2 = S_2 + S_{12} + S_{23} + S_{123}. \tag{7.4.18}$$

By construction, $T_k \geq S_k$, $k = 1, \ldots, d$. The difference $T_k - S_k$ measures the influence of x_k due to its interactions with other variables. From (7.4.17), the calculation of T_k appears to require the determination of a total of $\sum_{j=0}^{d-1} \binom{d-1}{j}$ variances, V_Q. However there is at least one method of making this computation more efficient which we describe next.

For arbitrary $Q \subset \{1, \ldots, d\}$, let

$$V_Q^u = Var\left[u_Q(\boldsymbol{X}_Q)\right] = Var\left[E\left[y(\boldsymbol{X}) \mid \boldsymbol{X}_Q\right]\right] \tag{7.4.19}$$

be the variance of the uncorrected effect. The quantity V_Q^u can be interpreted as the average reduction in uncertainty in $y(\boldsymbol{x})$ when $\boldsymbol{x}_Q$ is fixed because

$$V_Q^u = Var\left[y(\boldsymbol{X})\right] - E\left[Var\left[y(\boldsymbol{X}) \mid \boldsymbol{X}_Q\right]\right] .$$

Consider two special cases of the uncorrected effect function variances. From (7.4.10), the variance of the uncorrected effect function $u_k(x_k)$ of the input x_k is

$$V_k^u = Var\left[y_k(X_k) + y_0\right] = V_k.$$

Thus the main effect sensitivity index of input x_k, S_k, can also be computed using

$$S_k = \frac{V_k^u}{V}. \tag{7.4.20}$$

Using (7.4.11) and the orthogonality property (7.4.14), the variance of the uncorrected effect $u_{kj}(x_k, x_j)$ is

$$V_{kj}^u = Var\left[y_k(X_k) + y_j(X_j) + y_{kj}(X_k, X_j) + y_0\right] = V_k + V_j + V_{kj} . \tag{7.4.21}$$

Equation (7.4.21) contains both the variance of the main effects and the variance of the interaction effect of inputs x_k and x_j. Thus $V_Q^u \neq V_Q$ when Q contains more than one input.

Equation (7.4.21) can be extended to arbitrary Q. We illustrate the usefulness of this expression by developing a formula for T_k where $k \in \{1, \ldots, d\}$ is fixed. *When used as a subscript*, let $-k$ denote the set $\{1, \ldots, d\} - \{k\}$; for example, $\boldsymbol{X}_{-k}$ is the vector of all components of $\boldsymbol{X}$ *except* X_k. Then

$$\begin{aligned}
V_{-k}^u &= Var\left[u_{-k}\left(\boldsymbol{X}_{-k}\right)\right] \\
&= Var\left[y_{1,2,\ldots,k-1,k+1,\ldots,d}\left(\boldsymbol{X}_{-k}\right) + \cdots + y_1\left(X_1\right) + y_2\left(X_2\right) + \cdots\right. \\
&\qquad \left. + y_{k-1}\left(X_{k-1}\right) + y_{k+1}\left(X_{k+1}\right) + \cdots + y_d\left(X_d\right) + y_0\right] \\
&= Var\left[\sum_{Q:\, k \notin Q} y_Q\left(\boldsymbol{X}_Q\right)\right]
\end{aligned}$$

$$= \sum_{Q:\, k \notin Q} V_Q \,. \tag{7.4.22}$$

Equation (7.4.22) is the sum of all V_Q components *not involving* the subscript k in the variance decomposition (7.4.16). Thus $V - V^u_{-k}$ is the sum of all V_Q components that *do involve* the input x_k. Hence T_k can be expressed as

$$T_k = \frac{V - V^u_{-k}}{V} \,. \tag{7.4.23}$$

Thus if one is interested in estimating the *d main effect* SIs and the *d total effect* SIs, $\{S_k\}_k$ and $\{T_k\}_k$, only $2d$ uncorrected effect variances (7.4.19) need be computed.

Example 7.3 *(Continued).* Recall the g-function with d inputs is

$$y(\boldsymbol{x}) = \prod_{k=1}^{d} \frac{|4x_k - 2| + c_k}{1 + c_k} \,. \tag{7.4.24}$$

The S_k and T_k SIs are calculated as follows. Using the fact that if $X \sim U(0,1)$,

$$\begin{aligned} Var\,[\,|4X - 2|\,] &= 16\, Var\,[\,|X - 1/2|\,] \\ &= 16 \left\{ E\left[(X - 1/2)^2\right] - (E\,[\,|X - 1/2|\,])^2 \right\} = 1/3 \,. \end{aligned}$$

Hence the total variance of $y(\boldsymbol{x})$ is

$$\begin{aligned} V &= Var\,[y(\boldsymbol{X})] \\ &= Var\left[\prod_{\ell=1}^{d} \frac{|4X_\ell - 2| + c_\ell}{1 + c_\ell}\right] \\ &= E\left[\prod_{\ell=1}^{d} \left(\frac{|4X_\ell - 2| + c_\ell}{1 + c_\ell}\right)^2\right] - 1 \\ &= \prod_{\ell=1}^{d} E\left[\left(\frac{|4X_\ell - 2| + c_\ell}{1 + c_\ell}\right)^2\right] - 1 \\ &= \prod_{\ell=1}^{d} \left\{ Var\left[\frac{|4X_\ell - 2| + c_\ell}{1 + c_\ell}\right] + 1 \right\} - 1 \\ &= \prod_{\ell=1}^{d} \left\{ \frac{1}{(1 + c_\ell)^2} Var\,[\,|4X_\ell - 2|\,] + 1 \right\} - 1 \\ &= \prod_{\ell=1}^{d} \left(\frac{1}{3(1 + c_\ell)^2} + 1 \right) - 1 \,. \end{aligned}$$

For $k = 1, \ldots, d$, the numerator of S_k is the variance of the first-order effect function $y_k(X_k)$

$$V_k = Var\left[y_k(X_k)\right] = Var\left[\frac{|4X_k - 2| + c_k}{1 + c_k}\right] = \frac{1}{3(1 + c_k)^2},$$

and hence

$$S_k = V_k/V = \frac{\frac{1}{3(1+c_k)^2}}{\prod_{\ell=1}^{d}\left(\frac{1}{3(1+c_\ell)^2} + 1\right) - 1}. \tag{7.4.25}$$

In a similar fashion, for fixed $k = 1, \ldots, d$, the uncorrected effect function

$$u_{-k}(\boldsymbol{x}_{-k}) = \prod_{\ell \neq k} \frac{|4X_\ell - 2| + c_\ell}{1 + c_\ell}$$

has variance

$$\begin{aligned} V_{-k}^{u} &= Var\left[u_{-k}(\boldsymbol{X}_{-k})\right] \\ &= Var\left[\prod_{\ell \neq k} \frac{|4X_\ell - 2| + c_\ell}{1 + c_\ell}\right] \\ &= \prod_{\ell \neq k}\left(\frac{1}{3(1 + c_\ell)^2} + 1\right) - 1 \end{aligned}$$

using algebra similar to that in the derivation of V. Hence, after some simplification,

$$T_k = \frac{V - V_{-k}^{u}}{V} = \frac{\left(\frac{1}{1+3(1+c_k)^2}\right)\prod_{\ell=1}^{d}\left(\frac{1}{3(1+c_\ell)^2} + 1\right)}{\prod_{\ell=1}^{d}\left(\frac{1}{3(1+c_\ell)^2} + 1\right) - 1}. \tag{7.4.26}$$

As a specific example, consider the $d = 2$ case illustrated in Fig. 7.6 where $c_1 = 0.25$ and $c_2 = 10.0$. Calculation of (7.4.25) and (7.4.26) give the values in Table 7.8.

k	S_k	T_k
1	0.9846	0.9873
2	0.0127	0.0154

Table 7.8 Main effect and total effect sensitivity indices for the function (7.4.24) when $d = 2$ and $(c_1, c_2) = (0.25, 10.0)$

The estimated S_k and T_k in Table 7.8 are interpreted as saying that (1) x_1 is a far more active input than x_2 and (2) there is virtually no interaction between x_1 and x_2 because $S_{12} = T_1 - S_1 = T_2 - S_2 = 0.0027$. Figure 7.6, which plots the joint effect function for this example, qualitatively verifies the correctness of the $k = 1$ and $k = 2$ rows of Table 7.8. For each $x_2^0 \in [0, 1]$, $\{y(x_1, x_2^0) : 0 \leq x_1 \leq 1\}$ has a v-shaped profile that is independent of x_2^0; for each $x_1^0 \in [0,1]$, $\{y(x_1^0, x_2) : 0 \le x_2 \le 1\}$ is a horizontal line with height depending on x_1^0. ♦

Example 7.4. This section concludes with an example that shows both the strength and weakness of trying to summarize the behavior of a potentially complicated function by (several) real numbers. Consider

$$y(x_1, x_2, x_3) = (x_1 + 1)\cos(\pi x_2) + 0x_3 = \cos(\pi x_2) + x_1 \cos(\pi x_2) + 0x_3$$

defined on $[0,1]^3$. The formula shows that $y(\boldsymbol{x})$ has a term depending only on x_2, an x_1 by x_2 "interaction" term, and does not depend on x_3. Figure 7.7 plots (x_1, x_2) versus $y(x_1, x_2, 0.5)$ (which is the same for any other $x_3 \in [0,1]$). Any reasonable measure of the sensitivity of $y(\boldsymbol{x})$ to its inputs should show that x_3 has *zero* influence on $y(\boldsymbol{x})$, while both x_1 and x_2 are influential.

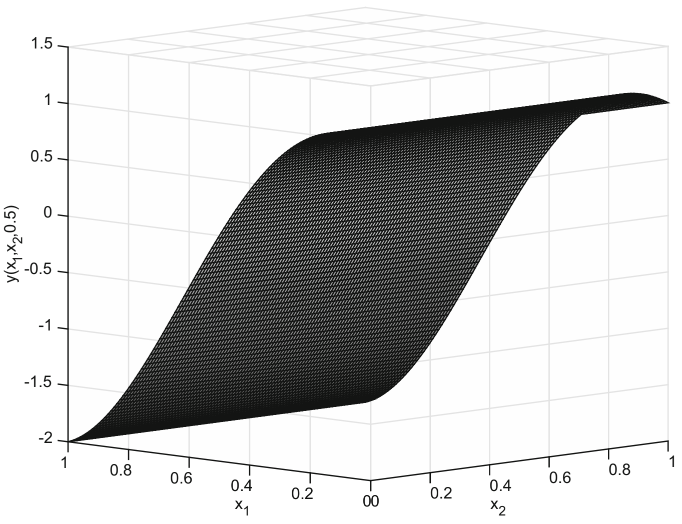


Fig. 7.7 The function $y(x_1, x_2, 0.5) = (x_1 + 1)\cos(\pi x_2)$ versus x_1 and x_2

Using the facts that

$$\int_0^1 \cos(\pi x)\, dx = 0 \text{ and } \int_0^1 \cos^2(\pi x)\, dx = \frac{1}{2},$$

it is straightforward to compute that the overall mean is

$$y_0 = \int_0^1 \int_0^1 \int_0^1 (x_1 + 1)\cos(\pi x_2)\, dx_1\, dx_2\, dx_3 = 0$$

and the uncentered effect functions are

$$u_1(x_1) = \int_0^1 \int_0^1 (x_1 + 1)\cos(\pi x_2)\, dx_2\, dx_3 = 0\,,$$

$$u_2(x_2) = \int_0^1 \int_0^1 (x_1 + 1)\cos(\pi x_2)\, dx_1\, dx_3 = \frac{3}{2}\cos(\pi x_2)\,, \text{ and}$$

$$u_3(x_3) = \int_0^1 \int_0^1 (x_1 + 1)\cos(\pi x_2)\, dx_1\, dx_2 = 0\,.$$

Because $y_0 = 0$, the centered effects $y_1(x_1)$, $y_2(x_2)$, and $y_3(x_3)$ are the same as $u_1(x_1)$, $u_2(x_2)$, and $u_3(x_3)$, respectively. That $y_3(x_3) = 0$ is expected, while $y_1(x_1) = 0$ may be unexpected. However, in this artificial example, for each fixed $(x_1, x_3) = (x_1^0, x_3^0)$, the function $y(x_1^0, x_2, x_3^0)$ is antisymmetric about $x_2 = 1/2$ and is constant with integral zero with respect to x_2. Letting $-Q$ denote the set $\{1, \ldots d\} \setminus Q$ in an extension of the $-k$ notation, any function $y(\boldsymbol{x})$ with constant average value over the inputs $\boldsymbol{x}_{-Q}$ has constant mean $u_Q(\boldsymbol{x}_Q)$ with respect to the inputs $\boldsymbol{x}_Q$.

Returning to the specifics of this example, the variance of $y(X_1, X_2, X_3)$ is

$$\begin{aligned} V &= Var\left[(X_1 + 1)\cos(\pi X_2)\right] \\ &= E\left[(X_1 + 1)^2\cos^2(\pi X_2)\right] = E\left[(X_1 + 1)^2\right] E\left[\cos^2(\pi X_2)\right] = \frac{7}{6}\,, \end{aligned}$$

which gives the main effect sensitivities

$$S_1 = \frac{Var\left[u_1(X_1)\right]}{V} = 0 = S_3$$

while

$$S_2 = \frac{Var\left[\frac{3}{2}\cos(\pi X_2)\right]}{V} = \frac{27}{28} \approx 0.964\,.$$

The zero main effect for x_1 is, perhaps, unexpected. It is due to the fact that the integral of $y(x_1, x_2)$ over the x_2-term is zero; any other function with a centered interaction term would also have $S_1 = 0$, e.g., $y(\boldsymbol{x}) = x_1(x_2 - 0.5)$. The large value of 0.964 for the main effect of x_2 also may not be consistent with the readers' intuition. This large value illustrates again that S_k depends on *every* x_k term that comprises the $y(\boldsymbol{x})$ formula, *not merely additive terms* $\beta \times x_i$.

To continue the example, we compute the total effects for each input using the formula (7.4.23). First, note that

$$u_{-1}(\boldsymbol{x}_{-1}) = u_{23}(x_2, x_3) = \int_0^1 y(x_1, x_2, x_3)\, dx_1 = 1.5\cos(\pi x_2)\,,$$

and similarly

$$u_{-2}(\boldsymbol{x}_{-2}) = u_{13}(x_1, x_3) = 0\,, \text{ and } u_{-3}(\boldsymbol{x}_{-3}) = u_{12}(x_1, x_2) = (x_1 + 1)\cos(\pi x_2)$$

so

$$V^u_{-1} = Var\,[1.5\ \cos(\pi X_2)] = 9/8, \quad V^u_{-2} = 0, \text{ and } V^u_{-3} = V = 7/6$$

yielding

$$T_1 = \frac{V - V^u_{-1}}{V} = \frac{6}{7}\left(\frac{7}{6} - \frac{9}{8}\right) = \frac{1}{28} \approx 0.036, \ T_2 = 1, \text{ and } T_3 = 0.$$

The result that $T_3 = 0$ implies that $S_{13} = 0 = S_{23} = S_{123}$ as one expects from the functional form of $y(\boldsymbol{x})$. Indeed the remaining interaction must be $S_{12} = T_1 - S_1 = T_2 - S_2 = 1/28$ from (7.4.18). This small value for the S_{12} interaction may, again, not be consistent with the reader's intuition but shows that once the main effect functions are subtracted from $u_{12}(x_1, x_2)$, there is very little variability in $y_{12}(x_1, x_2)$.

Indeed, it is interesting to note that for this example the variances of the centered and uncentered functions $y_{12}(x_1, x_2)$ and $u_{12}(x_1, x_2)$, respectively, can be quite different. In this case calculation gives

$$y_{12}(x_1, x_2) = (x_1 - 0.5)\cos(\pi x_2)$$

so that $V_{12} = Var[y_{12}(X_1, X_2)] = 1/24 \ll 7/6 = Var[u_{12}(X_1, X_2)] = V^u_{12}$. In general, the 2-$d$ sensitivity index for inputs x_i and x_j, S_{ij}, subtracts the associated main effect functions which can greatly reduce the variance of the averaged function values. ♦

7.5 Estimating Effect Plots and Global Sensitivity Indices

For simplicity, this section will assume that the input space is rectangular and has been scaled and shifted to $[0, 1]^d$. The section will describe how quadrature, empirical (plug-in) Bayesian, and fully Bayesian methods can be used to estimate effect plots and *main effect* and *total effect* sensitivity indices based on training data, $(\boldsymbol{x}_i, y(\boldsymbol{x}_i))$, $i = 1, \ldots, n_s$. With one exception, these methods assume that $y(\boldsymbol{x})$ can be modeled as a realization of the regression plus stationary GP,

$$Y(\boldsymbol{x}) = \sum_{(\ell_1,\ldots,\ell_d)\in\mathcal{L}} \beta_{\ell_1\ldots\ell_d} \prod_{j=1}^{d} x_j^{\ell_j} + Z(\boldsymbol{x}), \tag{7.5.1}$$

where $\boldsymbol{\beta} = \{\beta_{\ell_1\ldots\ell_d}\}_{\ell_1\ldots\ell_d}$ is the vector of regression coefficients (ordered, say, lexicographically), the powers $\ell_1, \ldots, \ell_d$ are specified nonnegative integers, $\mathcal{L}$ is the collection of d-tuples corresponding to each term in the regression model, and $Z(\boldsymbol{x})$ is a zero mean stationary GP with process variance σ^2_Z and separable correlation

$$Cor\,[Z(\boldsymbol{x}_r), Z(\boldsymbol{x}_s)] = \prod_{j=1}^{d} R\left(x_{r,j} - x_{s,j} \,\middle|\, \boldsymbol{\kappa}_j\right), \tag{7.5.2}$$

for $1 \leq r, s \leq n_s$, which is defined by the parametric correlation function $R(\cdot \mid \cdot)$. Here $\boldsymbol{\kappa}_j$ is the correlation parameter (possibly a vector) associated with the j^{th} input; $\boldsymbol{\kappa} = (\boldsymbol{\kappa}_1, \ldots, \boldsymbol{\kappa}_d)$ denotes the vector of all the correlation parameters.

Note that the mean of the $Y(\boldsymbol{x})$ process in (7.5.1) is of the regression form

$$\sum_{j=1}^{p} \beta_j f_j(\boldsymbol{x}) = \sum_{(\ell_1,\ldots,\ell_d)\in\mathcal{L}} \beta_{\ell_1\ldots\ell_d} \prod_{j=1}^{d} x_j^{\ell_j}$$

where, for simplicity, the regression functions are polynomials. More complicated regression functions can be used in the estimators derived below, but formulas for many of these cases have not been worked out.

For example, the methods described below can be applied to both the Gaussian correlation function,

$$R_G(h \mid \xi) = \exp\left\{-\xi\, h^2\right\}, \tag{7.5.3}$$

where $\xi > 0$, and the cubic correlation function

$$R_C(h \mid \psi) = \begin{cases} 1 - 6\left(\frac{h}{\psi}\right)^2 + 6\left(\frac{|h|}{\psi}\right)^3, & |h| \leq \frac{\psi}{2} \\ 2\left(1 - \frac{|h|}{\psi}\right)^3, & \frac{\psi}{2} < |h| \leq \psi \\ 0, & \psi < |h| \end{cases}, \tag{7.5.4}$$

where $\psi > 0$. The estimated effect plots and sensitivity indices can also be derived for the separable Bohman, the separable Matern, and the separable power exponential correlation functions using the methods described below.

7.5.1 Estimating Effect Plots

Given output function $y(\boldsymbol{x}) = y(x_1, \ldots, x_d)$, recall that for fixed $k \in \{1, \ldots, d\}$, the main effect (ME) plot for input x_k displays $(x_k, u_k(x_k))$, $0 \leq x_k \leq 1$, where

$$\begin{aligned} u_k(x_k) &= \int_0^1 \cdots \int_0^1 y(x_1, \ldots, x_d) \prod_{\ell \neq k} dx_\ell \\ &= \int_0^1 \cdots \int_0^1 y(x_1, \ldots, x_d)\, d\boldsymbol{x}_{-k} = E\left[y(\boldsymbol{X}) \mid X_k = x_k\right]. \end{aligned} \tag{7.5.5}$$

Here $u_k(x_k)$ is the average value of $y(\boldsymbol{x})$ when the k^{th} input is fixed at x_k and the averaging is over all possible values for inputs x_j, $j \neq k$. More generally, one can examine changes in the average value of $y(\boldsymbol{x})$ when two (or more) inputs are fixed in a 3-D joint effect (JE) plot of $(x_k, x_j, u_{kj}(x_k, x_j))$, $0 \leq x_k, x_j \leq 1$, where

$$u_{kj}(x_k, x_j) = \int_0^1 \cdots \int_0^1 y(x_1, \ldots, x_d) \prod_{\ell \neq k,j} dx_\ell \,.$$

Two methods of predicting $u_k(x_k)$ based on training data will be described. The first uses *quadrature*, and the second is a *Bayesian* predictor. Both methods can be extended to estimate joint effect or higher-effect function(s).

Naive quadrature estimates $u_k(x_k)$ using the definition of the integral to form

$$\widehat{u}_k(x_k) = \sum_{\Delta^\star} \widehat{y}(x_1^\star, \ldots, x_{k-1}^\star, x_k, x_{k+1}^\star, \ldots, x_d^\star)\; Vol(\Delta^\star), \qquad 0 \leq x_k \leq 1, \tag{7.5.6}$$

where $\widehat{y}(\boldsymbol{x})$ is a predictor of $y(\boldsymbol{x})$ and the sum is over a set of disjoint hyperrectangles $\Delta^\star$ that partition the $(x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_d)$ domain and $(x_1^\star, \ldots, x_{k-1}^\star, x_{k+1}^\star, \ldots, x_d^\star) \in \Delta^\star$. In principle $\widehat{y}(\boldsymbol{x})$ can be *any predictor* of $y(\boldsymbol{x})$; for example, $\widehat{y}(\boldsymbol{x})$ can be based on ordinary least squares regression, a neural net, or one of the kriging predictors described in Chap. 3 where the latter depends on what is known about the model parameters. Of course, the accuracy of $\widehat{u}_k(x_k)$ depends on the accuracy of $\widehat{y}(\boldsymbol{x})$ and the number of volume elements.

A kriging predictor of $y(\boldsymbol{x})$ is appropriate to use in (7.5.6) if $y(\boldsymbol{x})$ can be described as a realization of a GP having form (7.5.1) with *separable* correlation function (7.5.2). Assume the most frequently occurring case in which $(\boldsymbol{\beta}, \sigma_Z^2, \boldsymbol{\kappa})$ is unknown. In this situation, recall that when viewed as a function of the input $\boldsymbol{x} = (x_1, \ldots, x_d)$, an EBLUP of $y(\boldsymbol{x})$ based on estimated correlation parameter $(\widehat{\boldsymbol{\kappa}}_1, \ldots, \widehat{\boldsymbol{\kappa}}_d)$ has the form

$$\widehat{y}(\boldsymbol{x}) = d_0(\boldsymbol{x}) + \sum_{i=1}^{n_s} d_i \prod_{j=1}^{d} R\left(x_j - x_{i,j} \,\middle|\, \widehat{\boldsymbol{\kappa}}_j\right) \tag{7.5.7}$$

where the coefficients $\{d_i\}_i$ are described below (3.2.18) and, in particular,

$$d_0(\boldsymbol{x}) = \sum_{(\ell_1,\ldots,\ell_d)\in\mathcal{L}} \widehat{\beta}_{\ell_1\ldots\ell_d} \prod_{j=1}^{d} x_j^{\ell_j}$$

with $\widehat{\boldsymbol{\beta}}$ elements that are obtained from the weighted least squares estimator of $\boldsymbol{\beta}$ using estimated parameters, say the REML of $\boldsymbol{\kappa}$. In this case

$$\begin{aligned}
\widehat{u}_k(x_k) &= \int_0^1 \cdots \int_0^1 \left(\sum_{(\ell_1,\ldots,\ell_d)\in\mathcal{L}} \widehat{\beta}_{\ell_1\ldots\ell_d} \prod_{j=1}^{d} x_j^{\ell_j} + \sum_{i=1}^{n_s} d_i \prod_{j=1}^{d} R\left(x_j - x_{i,j} \,\middle|\, \widehat{\boldsymbol{\kappa}}_j\right) \right) d\boldsymbol{x}_{-k} \\
&= \sum_{(\ell_1,\ldots,\ell_d)\in\mathcal{L}} \widehat{\beta}_{\ell_1\ldots\ell_d}\, x_k^{\ell_k} \prod_{j\neq k} \left(\ell_j + 1\right)^{-1}
\end{aligned}$$

$$+\sum_{i=1}^{n_s} d_i\, R\left(x_k - x_{i,k} \,\middle|\, \widehat{\kappa}_k\right) \prod_{j\neq k} \int_0^1 R\left(x_j - x_{i,j} \,\middle|\, \widehat{\kappa}_j\right)\, dx_j, \quad 0 \le x_k < 1. \tag{7.5.8}$$

In some cases the one-dimensional integrals in (7.5.8) have a closed-form expression. For example, for the Gaussian correlation function (7.5.3),

$$\begin{aligned}\int_0^1 R\left(x_j - x_{i,j} \,\middle|\, \widehat{\kappa}_j\right) dx_j &= \int_0^1 \exp\left\{-\widehat{\kappa}_j\,(x_j - x_{i,j})^2\right\} dx_j \\ &= \frac{\sqrt{\pi}}{\sqrt{\widehat{\kappa}_j}} \left\{\Phi\left((1 - x_{i,j})\sqrt{2\widehat{\kappa}_j}\right) - \Phi\left((0 - x_{i,j})\sqrt{2\widehat{\kappa}_j}\right)\right\}\end{aligned}$$

where $\Phi(\cdot)$ is the standard normal cumulative distribution function (see Chen et al. (2005)). Svenson et al. (2014) provide closed-form expressions for (7.5.8) for the cubic (7.5.4) and Bohman correlation functions (2.2.14).

The idea of the *Bayesian estimator* of $u_k(x_k)$ is to *replace* $y(\bullet)$ by $Y(\bullet)$ in the integral (7.5.5) that defines $u_k(x_k)$ yielding a process $U_k(x_k)$. Then the posterior estimate $u_k(x_k)$ is the conditional mean of $U_k(x_k)$ given the data. In more detail, fix $x_k \in [0, 1]$. This estimator is obtained by observing that, conditionally given model parameters, the integral

$$U_k(x_k) = \int_0^1 \cdots \int_0^1 Y(x_1, \ldots, x_d) \prod_{\ell\neq k} dx_\ell = E_{X_{-k}}\left[Y(\boldsymbol{X}) \mid X_k = x_k\right] \tag{7.5.9}$$

of the GP $Y(\boldsymbol{x})$ is a GP under mild assumptions (see Yaglom (1962) or Adler (1990)). The subscript $\boldsymbol{X}_{-k}$ in $E_{X_{-k}}[\cdot]$ denotes averaging with respect to the i.i.d. $U(0, 1)$ distributions of the $\{X_\ell\}_{\ell\neq k}$. Intuitively (7.5.9) is a GP because $U_k(\boldsymbol{x}_k)$ is approximately a linear combination of $Y(\boldsymbol{x})$ values and a linear combination of multivariate normal random variables has a multivariate normal distribution.

Used below, the (conditional) mean, variance, and covariance of $U_k(x_k)$ given the model parameters can be obtained by interchanging appropriate integrals as follows. The mean of $U_k(x_k)$ is

$$\begin{aligned}E_Y\left[U_k(x_k)\right] &= E_Y\left[E_{X_{-k}}\left[Y(\boldsymbol{X}) \mid X_k = x_k\right]\right] \\ &= E_{X_{-k}}\left[\sum_{(\ell_1,\ldots,\ell_d)\in\mathcal{L}} \beta_{\ell_1\ldots\ell_d} \prod_{j=1}^{d} x_j^{\ell_j} \,\middle|\, X_k = x_k\right] \\ &= \sum_{(\ell_1,\ldots,\ell_d)\in\mathcal{L}} \beta_{\ell_1\ldots\ell_d}\, x_k^{\ell_k} \prod_{j\neq k} \int_0^1 x_j^{\ell_j} dx_j \\ &= \sum_{(\ell_1,\ldots,\ell_d)\in\mathcal{L}} \beta_{\ell_1\ldots\ell_d}\, x_k^{\ell_k} \prod_{j\neq k} \left(\ell_j + 1\right)^{-1}.\end{aligned} \tag{7.5.10}$$

When $E_Y[Y(\boldsymbol{x})] = \beta_0$, (7.5.10) simplifies to $E_Y[U_k(x_k)] = \beta_0$. The conditional covariance of $U_k(x_k)$ can be calculated as follows for an arbitrary separable correlation function, $R(\cdot \mid \boldsymbol{\kappa})$. Let $\boldsymbol{x}_i = (x_{i,1}, \ldots, x_{i,k-1}, x_{i,k}, x_{i,k+1}, \ldots, x_{i,d})$ for $i = 1, 2$, then

$$\begin{aligned} Cov_Y\left[U_k(x_{1,k}), U_k(x_{2,k})\right] &= \\ & Cov_Y\left[\int \cdots \int Y(\boldsymbol{x}_1) \prod_{j \neq k} dx_{1,j}, \int \cdots \int Y(\boldsymbol{x}_2) \prod_{j \neq k} dx_{2,j}\right] \\ &= \sigma_Z^2 \int \cdots \int R\left(\boldsymbol{x}_1, \boldsymbol{x}_2 \mid \boldsymbol{\kappa}\right)\, d\boldsymbol{x}_{1,-k}\, d\boldsymbol{x}_{2,-k} \\ &= \sigma_Z^2 R\left(x_{1,k}, x_{2,k} \mid \boldsymbol{\kappa}_k\right) \prod_{j \neq k}\left[\int_0^1 \int_0^1 R\left(x_{1,j}, x_{2,j} \mid \boldsymbol{\kappa}_j\right)\, dx_{1,j}\, dx_{2,j}\right], \end{aligned} \tag{7.5.11}$$

with the process variance of $U_k(x_k)$ being the special case

$$\sigma_U^2 \equiv Cov_Y\left[U_k(x_k), U_k(x_k)\right] = \sigma_Z^2 \prod_{j \neq k}\left[\int_0^1 \int_0^1 R\left(x_{1,j}, x_{2,j} \mid \boldsymbol{\kappa}_j\right) dx_{1,j}\, dx_{2,j}\right]. \tag{7.5.12}$$

Closed-form formulas for $Cov_Y[U_k(x_{1,k}), U_k(x_{2,k})]$ are known for the Gaussian, cubic, and Bohman correlation functions (Chen et al. (2005, 2006); Svenson et al. (2014)).

Returning to the description of the Bayesian predictor of $u_k(x_k)$, suppose for *the remainder of this section*, that the GP (7.5.1) is

$$Y(\boldsymbol{x}) = \beta_0 + Z(\boldsymbol{x}),$$

given $(\beta_0, \sigma_Z^2, \boldsymbol{\kappa})$ and thus $Y(\boldsymbol{x})$ has conditionally the mean β_0. As usual, let $\boldsymbol{y}^{n_s} = (y(\boldsymbol{x}_1), \ldots, y(\boldsymbol{x}_{n_s}))^\top$ denote the training data which are observed at inputs $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n_s}$ and that $\boldsymbol{Y}^{n_s}$ is the corresponding model vector.

It can be shown that given model parameters, $(U_k(x_k), Y(\boldsymbol{x}_1), \ldots, Y(\boldsymbol{x}_{n_s}))$ has the joint multivariate normal distribution

$$\begin{pmatrix} U_k(x_k) \\ \boldsymbol{Y}^{n_s} \end{pmatrix} \sim N_{1+n_s}\left(\begin{pmatrix} \beta_0 \\ \boldsymbol{1}_{n_s}\beta_0 \end{pmatrix}, \begin{bmatrix} \sigma_U^2 & \boldsymbol{\Sigma}_{Un_s} \\ \boldsymbol{\Sigma}_{n_sU} & \boldsymbol{\Sigma}_{n_sn_s} \end{bmatrix}\right),$$

where all components of the covariance can be calculated using methods similar to those in (7.5.10)–(7.5.12) and are known because $(\beta_0, \sigma_Z^2, \boldsymbol{\kappa})$ is known. In particular, the variance σ_U^2 is given by (7.5.12); the cross-covariance $\boldsymbol{\Sigma}_{Un_s}$ is the $1 \times n_s$ vector with i^{th} component $Cov_Y[U_k(x_k), Y(\boldsymbol{x}_i)]$, $1 \leq i \leq n_s$; $\boldsymbol{\Sigma}_{n_sU} = \boldsymbol{\Sigma}_{Un_s}^\top$; and $\boldsymbol{\Sigma}_{n_sn_s}$ is the $n_s \times n_s$ matrix of variances and covariances of $\boldsymbol{Y}^{n_s}$. By interchanging expectations, the cross-covariance can be calculated for separable correlation functions (7.5.2) as

$$Cov_Y\left[U_k(x_k), Y(\boldsymbol{x}_1)\right] = Cov_Y\left[\int \cdots \int Y(\boldsymbol{x})\, d\boldsymbol{x}_{-k},\ Y(\boldsymbol{x}_1)\right]$$
$$= \sigma_Z^2 R\left(x_k - x_{1,k} \mid \kappa_k\right) \prod_{j \neq k} \int R\left(x_j - x_{1,j} \mid \kappa_j\right) dx_j$$

at generic inputs $\boldsymbol{x}_1 = (x_{1,1}, \ldots, x_{1,d})$ and $x_k \in [0, 1]$. Formulas for $Cov_Y[U_k(x_k), Y(\boldsymbol{x}_1)]$ are known for the Gaussian, cubic, and Bohman correlation functions (Chen et al. (2005, 2006); Svenson et al. (2014)). Thus the (posterior mean) Bayes estimator of $u_k(x_k)$ is

$$\widehat{u}_k(x_k) = E_Y\left[U_k(x_k) \,\middle|\, \boldsymbol{Y}^{n_s}, (\beta_0, \sigma_Z^2, \boldsymbol{\kappa})\right]$$

which is

$$\widehat{u}_k(x_k) = \beta_0 + \boldsymbol{\Sigma}_{Un_s} \boldsymbol{\Sigma}_{n_s n_s}^{-1} \left(\boldsymbol{Y}^{n_s} - \boldsymbol{1}_{n_s}\beta_0\right). \tag{7.5.13}$$

The empirical Bayes estimator of $u_k(x_k)$ plugs estimated parameters into the formula (7.5.13) for $\widehat{u}_k(x_k)$. *When the parameters* $(\beta_0, \sigma_Z^2, \boldsymbol{\kappa})$ *can be described by a prior distribution* $[\beta_0, \sigma_Z^2, \boldsymbol{\kappa}]$ then

$$\begin{aligned}\widehat{u}_k(x_k) &= E_Y\left[U_k(x_k) \mid \boldsymbol{Y}^{n_s}\right] \\ &= E_{[(\beta_0,\sigma_Z^2,\boldsymbol{\kappa}) \mid \boldsymbol{Y}^{n_s}]} E_Y\left[U_k(x_k) \,\middle|\, \boldsymbol{Y}^{n_s}, (\beta_0, \sigma_Z^2, \boldsymbol{\kappa})\right] \\ &= E_{[(\beta_0,\sigma_Z^2,\boldsymbol{\kappa}) \mid \boldsymbol{Y}^{n_s}]}\left[\beta_0 + \boldsymbol{\Sigma}_{Un_s} \boldsymbol{\Sigma}_{n_s n_s}^{-1} \left(\boldsymbol{Y}^{n_s} - \boldsymbol{1}_{n_s}\beta_0\right)\right]\end{aligned}$$

where the (outer) expectation is with respect to the posterior of the parameters given the calculated output data. In practice, $\widehat{u}_k(x_k)$ is approximated by averaging (7.5.13) over draws from the $(\beta_0, \sigma_Z^2, \boldsymbol{\kappa})$ posterior distribution (see Oakley (2009), Moon (2010), Svenson (2011)).

Example 7.3 (Continued). Consider the g-function (7.4.5) with $d = 4$ and $\boldsymbol{c} = (0.1, 1.0, 2.0, 5.0)$. Recall that the k^{th} main effect function of the g-function is

$$u_k(x_k) = \frac{|4x_k - 2| + c_k}{1 + c_k}.$$

Figure 7.8 shows the ME plots $u_1(x_1)$, $u_2(x_2)$, $u_3(x_3)$, and $u_4(x_4)$ for this example. The activity of the inputs is inversely ordered according to their c_i values so that x_1, corresponding to $c_1 = 0.1$, is the most active and x_4, corresponding to $c_4 = 5.0$, is least active.

Suppose that $n_s = 40$ points are selected in $[0, 1]^4$ according to a MmLHD. Figure 7.9 plots the empirical version of (7.5.13) based on a GP with Gaussian correlation function and REML covariance parameter estimates. The estimated $u_k(x_k)$, $k = 1, \ldots, 4$, are qualitatively consistent with the true main effect functions; however they differ in two respects. First, they do not capture the extreme behavior of the true ME functions at the endpoints and center of the input; this is to be expected because kriging estimators tend to not capture the extreme behavior of the true $y(\boldsymbol{x})$. Second, the estimated $u_k(x_k)$ is rounded at its minimum which is caused by the fact the Gaussian correlation structure produces estimates that are infinitely

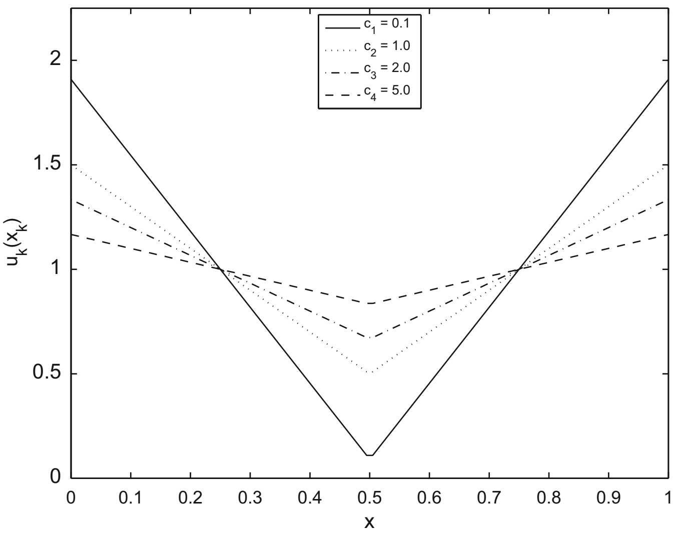


Fig. 7.8 Main Effect Plots of $u_1(x_1)$, $u_2(x_2)$, $u_3(x_3)$, and $u_4(x_4)$ for the g-function when $\boldsymbol{c} = (0.1, 1.0, 2.0, 5.0)$

differentiable at all inputs x_k. The true $u_k(x_k)$ is continuous but not differentiable at its minimizer $x_k = 1/2$. ♦

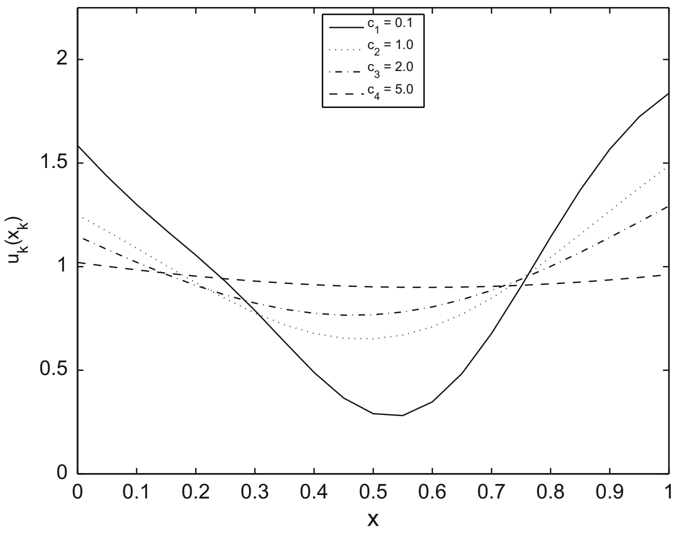


Fig. 7.9 Empirical Bayes estimated main effect plots for $u_1(x_1)$, $u_2(x_2)$, $u_3(x_3)$, and $u_4(x_4)$ for Example 7.3 based on REML correlation parameter estimates obtained from a 40-point MmLHD

Example 1.3 *(Continued)*. As a final example, the estimated main effect plots are presented for the eight inputs of the borehole function

$$y(\boldsymbol{x}) = \frac{2\pi T_u(H_u - H_l)}{\ell n(r/r_w)\left[1 + \frac{2LT_u}{\ell n(r/r_w) r_w^2 K_w} + \frac{T_u}{T_l}\right]}.$$

The inputs of the borehole function and their ranges are listed in Table 7.5.

Recall that examination of the EEs based on a 45 run design containing 5 tours showed that the annual flow rate increased greatly in r_w, increased (at a slower rate) in H_u, and decreased in H_l and L.

Figure 7.10 shows that annual flow rate increases by about 140 m^3/year as r_w increases from 0.05 to 0.15; the annual flow rate increases by roughly 30 m^3/year as H_u increases from 990 to 1110; and it decreases by about 30 m^3/year as H_l and L increase over their ranges. The ME plots do not provide information about interaction among the inputs, although joint effect plots would allow an assessment of input interaction.

A useful addition to the ME plot is to show the range of annual flow rates for *fixed values of one or more inputs*. To illustrate, consider r_w, the input having the greatest influence on annual flow rate. Fix a grid of 20 equally spaced r_w values; for each given $r_w^\star$ in the grid, Fig. 7.11 displays side-by-side boxplots of the estimated $y(r, r_w^\star, T_u, \ldots, K_w)$ values when the seven remaining inputs, $(r, T_u, \ldots, K_w)$, vary over their ranges. This plot is constructed by, first, determining the EBLUP of $y(r, r_w, T_u, \ldots, K_w)$ and, second, by evaluating the EBLUP at a set of $(r, r_w^\star, T_u, \ldots, K_w)$ where the seven unconstrained inputs are selected according

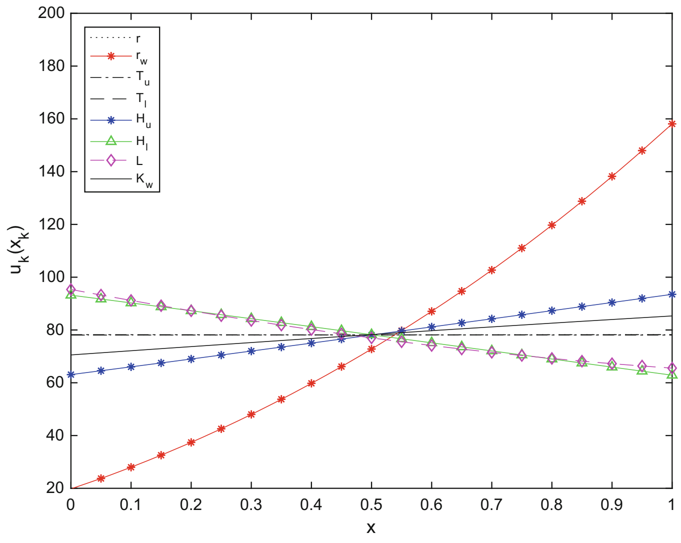


Fig. 7.10 Estimated main effect plots of $u_1(x_1), \ldots, u_8(x_8)$ for the borehole function using the empirical version of (7.5.13) based on REML correlation parameter estimates

to a space-filling design. In this example, the R package `slhd` of Ba (see Ba et al. (2015)) was used to select an approximate 70×7 maximin Latin hypercube design (based on the ten samples/input rule of thumb). The medians of each boxplot are joined by line segments. The piecewise linear curve verifies that the estimated ME curve at each fixed $r_w^\star$ is indeed the average of $y(r, r_w^\star, T_u, \ldots, K_w)$ over the remaining inputs. More importantly, it shows how the conditional uncertainty in $y(r, r_w, T_u, \ldots, K_w)$ increases greatly as r_w increases.

Contrast the conditional annual flow rate given r_w with that given H_l. Recall that the ME plot shows that the annual flow rate decreases moderately as H_l increases. Figure 7.12 displays the analogous side-by-side boxplots of estimated annual flow rates conditional on fixed H_l values. The trend in the medians clearly shows a decrease in the annual flow rate, verifying the Fig. 7.10 ME plot for H_l. In addition, a decrease in variability in the conditional annual flow rate is clearly visible. Furthermore the decrease in variability is seen to be caused by the fact that large annual flow rates are less possible as H_l increases. ♦

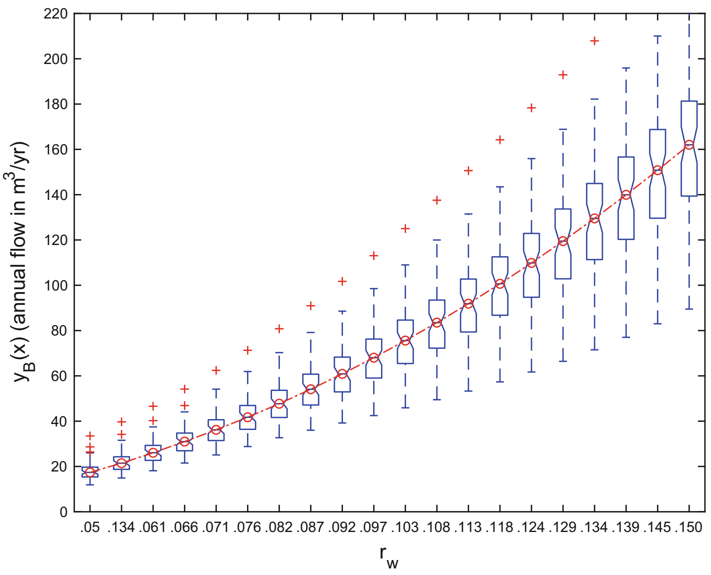


Fig. 7.11 Estimated range of the annual flow rate conditional on r_w, for 20 r_w inputs. The joined medians of the boxplots reproduce the estimated ME curve for r_w that is given in Fig. 7.10

7.5.2 Estimating Global Sensitivity Indices

Analogous to the first method described in Sect. 7.5.1, one estimator of S_k and T_k in (7.4.20) and (7.4.23), respectively, can be obtained by replacing $y(\boldsymbol{x})$ in the variance expressions V, V_k^u, and V_{-k}^u, $k = 1, \ldots, d$, by a predictor $\widehat{y}(\boldsymbol{x})$. To illustrate,

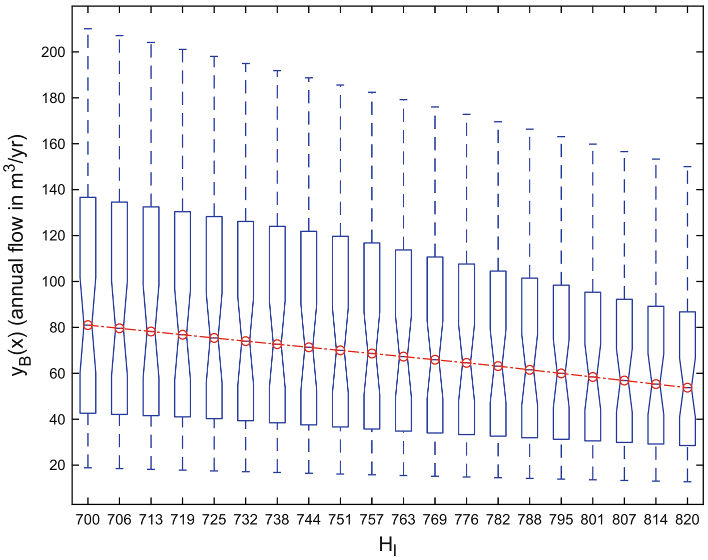


Fig. 7.12 Estimated range of annual flow rate conditional on H_l, for 20 H_l inputs. The joined medians of the boxplots reproduce the estimated ME curve for H_l that is given in Fig. 7.10

consider estimating the simplest of these quantities, the total variance

$$V = Var\,[y(\boldsymbol{X})] = E\left[y^2\,(\boldsymbol{X})\right] - \left(E\,[y\,(\boldsymbol{X})]\right)^2 = E\left[y^2\,(\boldsymbol{X})\right] - y_0^2$$

which can be obtained by estimating y_0 and $E[y^2(\boldsymbol{X})]$. To estimate y_0, substitute (7.5.7) into the definition of y_0 to obtain

$$\begin{aligned}
\widehat{y}_0 &= \int_0^1 \cdots \int_0^1 \widehat{y}(\boldsymbol{x})\, d\boldsymbol{x} \\
&= d_0 + \sum_{i=1}^{n_s} d_i \int_0^1 \cdots \int_0^1 \prod_{j=1}^{d} R\left(x_j - x_{i,j} \,\middle|\, \widehat{\boldsymbol{\kappa}}_j\right) \prod_{j=1}^{d} dx_j \\
&= d_0 + \sum_{i=1}^{n_s} d_i \prod_{j=1}^{d} \int_0^1 R\left(x_j - x_{i,j} \,\middle|\, \widehat{\boldsymbol{\kappa}}_j\right) dx_j
\end{aligned} \tag{7.5.14}$$

where

$$d_0 = \sum_{(\ell_1,\ldots,\ell_d)\in\mathcal{L}} \widehat{\beta}_{\ell_1\ldots\ell_d} \prod_{j=1}^{d} \left(\ell_j + 1\right)^{-1} .$$

This same one-dimensional integral in (7.5.14) occurred in the previous section in (7.5.8) and, as noted there, will have a closed-form expression for certain correlation functions. The plug-in estimate of $E[y^2(\boldsymbol{X})]$ is

$$E\left[\widehat{y}^2(\boldsymbol{X})\right] = \int_0^1 \cdots \int_0^1 \left(d_0(\boldsymbol{x}) + \sum_{i=1}^{n_s} d_i \prod_{j=1}^{d} R\left(x_j - x_{i,j} \,\middle|\, \widehat{\boldsymbol{\kappa}}_j\right) \right)^2 d\boldsymbol{x} \,. \tag{7.5.15}$$

The squared integrand in (7.5.15) is

$$\begin{aligned} &d_0^2(\boldsymbol{x}) + \sum_{i=1}^{n_s} d_i^2 \prod_{j=1}^{d} R^2\left(x_j - x_{i,j} \,\middle|\, \widehat{\boldsymbol{\kappa}}_j\right) \\ &\quad + 2\, d_0(\boldsymbol{x}) \sum_{i=1}^{n_s} d_i \prod_{j=1}^{d} R\left(x_j - x_{i,j} \,\middle|\, \widehat{\boldsymbol{\kappa}}_j\right) \\ &\quad + 2 \sum_{1 \le h < i \le n_s} d_h\, d_i \prod_{j=1}^{d} R\left(x_j - x_{h,j} \,\middle|\, \widehat{\boldsymbol{\kappa}}_j\right) R\left(x_j - x_{i,j} \,\middle|\, \widehat{\boldsymbol{\kappa}}_j\right) . \end{aligned} \tag{7.5.16}$$

The integrals of the terms in (7.5.16) can be expressed as products of one-dimensional integrals. Similar, though more complicated expressions can be derived for estimating V_k^u and V_{-k}^u, $k = 1, \ldots, d$.

Process-based estimators can also be formed for S_k and T_k, $1 \le k \le d$, using either fully Bayesian or empirical/plug-in Bayesian methods to handle unknown parameters (see Oakley (2009), Moon (2010), Svenson (2011)). In outline, the method is as follows.

The values of V_k^u and V_{-k}^u in (7.4.20) and (7.4.23) that are required to estimate S_k and T_k can be obtained from

$$V_Q^u = Var\left[E\left[Y(\boldsymbol{X}) \mid \boldsymbol{X}_Q\right]\right]$$

where $Q = \{k\}$. The inner expectation is over $\boldsymbol{X}_{-Q}$, and the outer variance is over $\boldsymbol{X}_Q$. The Bayesian estimator of V_Q^u is the posterior mean of V_Q^u given the observed code runs $\boldsymbol{Y}^{n_s}$; that is,

$$\widehat{V_Q^u} = E_Y\left[\, V_Q^u \,\middle|\, \boldsymbol{Y}^{n_s} \right] \tag{7.5.17}$$

where $E_Y[\cdot \mid \boldsymbol{Y}^{n_s}]$ denotes the conditional expectation of V_Q^u, a function of the $Y(\boldsymbol{x})$ process, given $\boldsymbol{Y}^{n_s}$.

Formulas for (7.5.17) are complicated but known for GP models $Y(\boldsymbol{x})$ having the form (7.5.1) with polynomial mean and separable Gaussian, Bohman, or cubic correlation function. They are also known if the observed data contains measurement error (see Chen et al. (2005)) and Svenson et al. (2014)).

Example 7.3 *(Continued)*. Recall from (7.4.25) that for the g-function (7.4.24),

$$S_k = \frac{\frac{1}{3(1+c_k)^2}}{\prod_{\ell=1}^{d}\left(\frac{1}{3(1+c_\ell)^2} + 1\right) - 1}$$

and

$$T_k = \frac{\left(\frac{1}{1+3(1+c_k)^2}\right)\prod_{\ell=1}^{d}\left(\frac{1}{3(1+c_\ell)^2}+1\right)}{\prod_{\ell=1}^{d}\left(\frac{1}{3(1+c_\ell)^2}+1\right)-1}.$$

These sensitivity indices are listed in Table 7.9 for the case $d = 4$ and $\boldsymbol{c} = (0.1, 1.0, 2.0, 5.0)$.

k	S_k	T_k
1	0.6174	0.7000
2	0.1868	0.2493
3	0.0830	0.1158
4	0.0208	0.0297

Table 7.9 ME and TE sensitivity indices for the function (7.4.24) when $d = 4$ and $\boldsymbol{c} = (0.1, 1.0, 2.0, 5.0)$

Using the same $n = 40$ points that were employed to construct the estimated main effect plots in Fig. 7.9, the sensitivity indices were estimated using both a plug-in REML estimate in (7.5.17) based on the constant mean, Gaussian correlation GP and its fully Bayesian analog. The results are listed in Table 7.10. In this case, a comparison with the true values for these training data shows that the plug-in predictor does a bit better than the fully Bayesian predictor by producing six of the eight predictors with smaller absolute errors. Of course no general conclusions can be deduced from this single example using one training data set. ♦

	Plug-in predictor		Fully Bayesian predictor	
k	$\widehat{S}_k$	$\widehat{T}_k$	$\widehat{S}_k$	$\widehat{T}_k$
1	0.6229	0.7431	0.5375	0.6857
2	0.1592	0.2620	0.1460	0.2825
3	0.0626	0.1522	0.0857	0.2330
4	0.0042	0.0536	0.0024	0.1128

Table 7.10 Estimated main effect and total effect sensitivity indices for the function (7.4.24) when $d = 4$ and $\boldsymbol{c} = (0.1, 1.0, 2.0, 5.0)$

Example 1.3 (Continued). The estimated EEs and ME plots for the eight inputs to the borehole function

$$y(\boldsymbol{x}) = \frac{2\pi T_u(H_u - H_l)}{\ell n(r/r_w)\left[1 + \frac{2LT_u}{\ell n(r/r_w)r_w^2K_w} + \frac{T_u}{T_l}\right]}$$

were presented earlier. Table 7.11 lists the estimated main effect and total effect sensitivity indices for each of the eight inputs.

From the ME sensitivity indices in Table 7.11, about 83% of the variability in the borehole function is due to changes in r_w; about 4% is due to individual changes in

Input	$\widehat{S}_k$	$\widehat{T}_k$
r	2.26e−06	6.05e−06
r_w	0.832	0.869
T_u	4.35e−06	7.97e−06
T_l	3.00e−06	5.81e−06
H_u	0.040	0.052
H_l	0.040	0.052
L	0.039	0.052
K_w	0.010	0.013

Table 7.11 Estimated main effect and total effect sensitivity indices for the eight inputs to the borehole function in Example 1.3

each of H_u, H_l, and L. The differences between the total SIs and the main effect SIs are small, and so there are no large interactions among the factors, although r_w does appear to have some small interactions with other inputs. ♦

Example 1.2 *(Continued).* Recall that in this example the simulator computes the failure depth for a sheet metal pocket formed under the manufacturing conditions described by six input variables (Montgomery and Truss (2001)). Figure 7.13 plots the failure depth versus each of the six inputs for the simulator data. These marginal plots suggest that *clearance* is the most important factor with larger clearances being associated with greater failure depths; *fillet radius* is less strongly but nevertheless positively associated with increasing failure depth, while none of the other inputs appears strongly associated with failure depth.

Applying the sensitivity tools introduced in this chapter, the main effect plots shown in Fig. 7.14 clearly show that increasing the *clearance* over its range raises the average failure depth over 150mm. Increasing the *fillet radius* or *punch plan view radius* over their ranges appears to increase the average failure depth by roughly 40mm. In contrast, the average failure depth is nearly constant in *lock bead distance*.

Table 7.12 lists the estimated main effect and total effect sensitivity indices for each of the six inputs. As anticipated, the input *clearance* explains, far and away, the greatest proportion of the variability in $y(\boldsymbol{x})$ with *fillet radius* and *punch plan view radius* being the next most important inputs. *Pocket width*, *pocket length*, and *lock bead distance* are unimportant. The differences between the total effect SIs and main effect SIs are all small, suggesting that there is minimal interaction among the inputs. ♦

7.6 Variable Selection

Because sophisticated computer simulators can have many inputs that describe, e.g., engineering design choices, environmental conditions, and uncertain model parameters, one important goal of the initial analysis of simulator output is the identification of those inputs which have a "significant" impact on the output. These variables are

Input	$\widehat{S}_k$	$\widehat{T}_k$
Clearance	0.7211	0.7657
Fillet radius	0.0881	0.1144
Punch plan view radius	0.0951	0.1276
Pocket width	0.0175	0.0385
Pocket length	0.0102	0.0429
Lock bead distance	0.0003	0.0057

Table 7.12 Estimated main effect and total effect sensitivity indices for the six inputs on the failure depth in Example 1.2

termed "active" inputs. The goal of identifying active inputs is termed *variable selection*. Variable selection is a decision-oriented objective that uses, among other tools, the sensitivity analysis methods described in Sects. 7.1–7.5.

This section describes two methods for performing variable selection based on simulator output $y(\boldsymbol{x})$ having inputs $\boldsymbol{x} = (x_1, \ldots, x_d)$. Both methods assume some knowledge is available about $y(\boldsymbol{x})$: either a prior is known for $y(\boldsymbol{x})$ (as in Chap. 4) or, more simply, the monotonicity of $y(\boldsymbol{x})$ as a function of each x_k is (roughly) obtainable. Linkletter et al. (2006) used a Bayesian analysis to identify active inputs given output from n_s simulator runs, $y(\boldsymbol{x}_1), \ldots, y(\boldsymbol{x}_{n_s})$; here $\boldsymbol{x}_i = (x_{i,1}, \ldots, x_{i,d})$, $i = 1, \ldots, n_s$. The method assumes a GP prior for $y(\boldsymbol{x})$. Moon et al. (2012) proposed a *two-stage design and analysis* methodology to identify active inputs which is es-

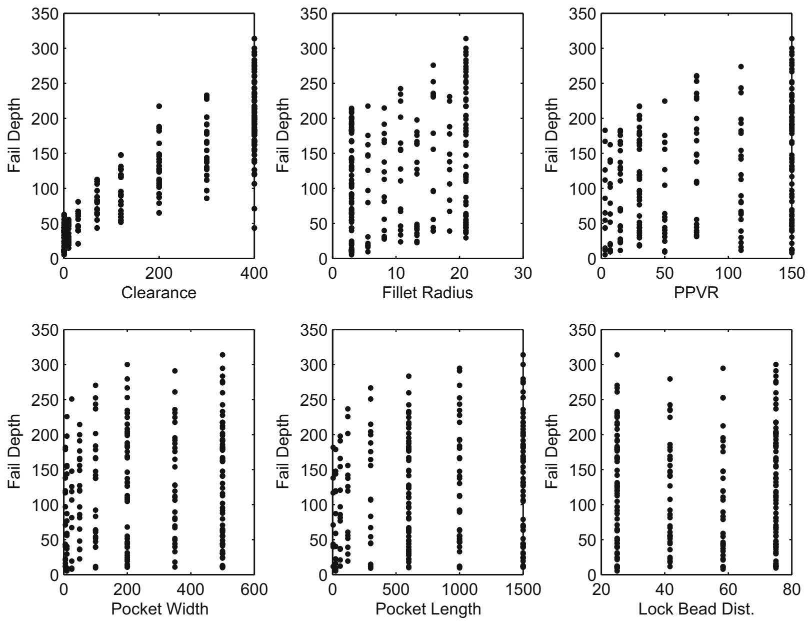


Fig. 7.13 Failure depth versus six potential input factors

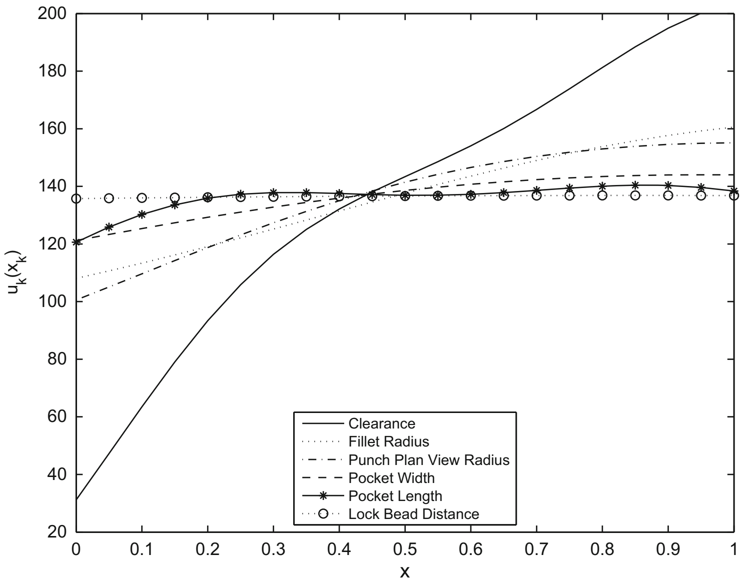


Fig. 7.14 Estimated main effect plots of the six input factors

pecially useful for cases with a large number of inputs d. The Moon et al method extended "group screening" methodology, originally developed for physical system experiments, to simulator output. The method assumes that either past experience with outputs similar to $y(\boldsymbol{x})$ or subject matter experts can inform the researcher of those inputs x_i which increase (and decrease) $y(\boldsymbol{x})$.

Using either the Linkletter et al. (2006) or Moon et al. (2012) procedures, once the active and low-active inputs have been identified, future runs of the simulator code can marginally sample the active inputs more comprehensively and the less active inputs more sparingly. Indeed, an investigator might use but one setting of a low-active input, say the center of its range.

The variable selection procedure of Linkletter et al. (2006) is based on the assumption that, after standardizing the simulator outputs $y(\boldsymbol{x}_1), \ldots, y(\boldsymbol{x}_{n_s})$ to have *sample mean zero* and *unit sample variance*, $y(\boldsymbol{x})$ can be regarded as a draw from a *zero mean stationary* GP with Gaussian covariance function

$$Cov\,[Y(\boldsymbol{x}_1), Y(\boldsymbol{x}_2)] = \lambda_Z^{-1} \prod_{k=1}^{d} \rho_k^{4\,(x_{1,k}-x_{2,k})^2} \,. \tag{7.6.1}$$

Linkletter et al. (2006) also recenter and rescale each input so that $x_{1,k}, \ldots, x_{n_s,k}$ satisfy $\min_i x_{i,k} = 0$ and $\max_i \; x_{i,k} = 1$, so that ρ_k can be interpreted as the correlation between $Y(\boldsymbol{x}_1)$ and $Y(\boldsymbol{x}_2)$ for any $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ that differ only in their k^{th} component by half the range of x_k, $k = 1, \ldots, d$.

The Linkletter et al. (2006) procedure is based on the observation that if ρ_k in (7.6.1) equals (or is near) 1 then $y(\boldsymbol{x})$ draws from the process must be (nearly) *functionally independent* of x_k, for $k = 1, \ldots, d$, i.e., x_k is an inert (low-active) input to $y(\boldsymbol{x})$. The method has two further key features. The first is the assumption that ρ_k has a prior of the form

$$[\rho_k] \sim \gamma\, U[0,1] + (1-\gamma)\, I\{\rho_k = 1\} \tag{7.6.2}$$

where $\gamma \in (0,1)$ and $I\{\rho_k = 1\}$ is the indicator function which equals 1 when $\rho_k = 1$ and is zero otherwise. Termed a "slab and spike" prior, the parameter γ in (7.6.2) is the prior probability that an input is *active* (Savitsky et al. (2011) and the references therein give additional uses of slab and spike priors in variable selection). Linkletter et al. (2006) chose γ to reflect "effect sparsity," meaning that a relatively low proportion of the inputs will be active; they take $\gamma = 0.25$ in examples. The second feature is their use of an inert input that creates a benchmark ("reference distribution") against which the correlations of the d inputs are compared. Moon et al. (2012) also use benchmark distributions, and hence this topic is discussed in more detail below.

Moon et al. (2012) adapt the research on group screening that has appeared in the physical experiment and stochastic simulation experiment literatures, to address deterministic computer simulator experiments (see Kleijnen (1987), Lewis and Dean (2001), Vine et al. (2005), Kleijnen et al. (2006), and Morris (2006) for recent work and reviews of the former). Stage 1 of the Moon et al. (2012) procedure uses the knowledge about the monotonicity of $y(\boldsymbol{x})$ to group similar-acting inputs $x_1, \ldots, x_d$. The initial set of runs varies all members of a group in a common manner. The procedure uses total effect sensitivity indices (TSIs) to identify the input *groups* that show low activity and eliminates them from further consideration. The remaining groups are assumed to contain one or more active inputs. Stage 2 samples the individual inputs in the active groups. It identifies *active inputs* by comparing the TSIs of each input to a benchmark TSI distribution created by adding a low noise input to the data.

In more detail, both Linkletter et al. (2006) and Moon et al. (2012) add an input, x_{d+1}, to the input data and analyze the $d+1$ input data

$$y^{\star}(\boldsymbol{x}, x_{d+1}) = y(\boldsymbol{x}) + \beta_{d+1} x_{d+1}\,, \tag{7.6.3}$$

where $x_{d+1} \in [0,1]$. In their examples, Linkletter et al. (2006) used an inert input, i.e., $\beta_{d+1} = 0$, to describe the benchmark activity level but mention the possibility of using a low-active input such as (7.6.4) with $\beta_{d+1} \neq 0$. They decide whether each input is active (or not) by forming multiple, say 500, data sets each with randomly selected $x_{d+1,1}, \ldots, x_{d+1,n}$ and taking a given number of draws, say 1000, from the posterior distribution of $\rho_1, \ldots, \rho_{d+1}$. For each $k \in \{1, \ldots, d\}$, they compare the median of the distribution of *all* posterior draws of ρ_k (combined over data sets) to an upper quantile, say the upper 10% quantile, of the 500 medians of the ρ_{d+1} posterior draws (one median from each data set). Input x_k is declared active if the the median of the ρ_k draws is *below* the selected quantile of the ρ_{d+1} medians.

Moon et al. (2012) take β_{d+1} to be a fraction of the range of the training data, i.e.,

$$\beta_{d+1} = \left(\max_{1\le i\le n} y(\boldsymbol{x}_i) - \min_{1\le i\le n} y(\boldsymbol{x}_i)\right) \times \tau\,, \tag{7.6.4}$$

where $\tau \in (0,1)$ is specified. Moon et al. (2012) performed a sensitivity and specificity study of their procedures' probability of correctly detecting high- and low-active inputs in a test bed of $y(\boldsymbol{x})$ and recommended $\tau = 0.14$ for their procedure. They construct their benchmark distribution by selecting the vectors $(x_{1,d+1}, \ldots, x_{n_s,d+1})$ to be orthogonal to the input vectors for the d input variables. In both stages they compare the TSI for each variable to that of the low-active input.

Example 1.2 *(Continued).* Recall that in this example the simulator computes the failure depth for a sheet metal pocket formed under the manufacturing conditions described by six input variables (see page 5). This example applied the Linkletter et al. (2006) variable selection procedure to the approximate $n_s = 60$ run training data set identified in Sect. 4.3.

The benchmark distribution for the posterior ρ draws was formed from the low-active input,

$$\begin{aligned}\left(\max_{1\le i\le 60} y(\boldsymbol{x}_i) - \min_{1\le i\le 60} y(\boldsymbol{x}_i)\right) &\times 0.14 \ \times x_{d+1}\\ &= (283.34 - 10.76) \times 0.14 \ \times x_7 = 38.16 \times x_7,\end{aligned} \tag{7.6.5}$$

which was added to the failure depth data. One hundred data sets, each of size $n_s = 60$, were formed by adding (7.6.5) to the original failure depth as in (7.6.3), creating $y^\star(\boldsymbol{x}, x_7)$ outputs with seven inputs.

The Linkletter et al procedure was applied to the 100 $y^\star(\boldsymbol{x}, x_7)$ data sets which were each centered and standardized to facilitate their modeling as draws from a zero mean GP having $\Gamma(5,5)$ prior for the process precision, and independent and identically distributed slab and spike priors (7.6.2) with $\gamma = 0.25$ for $\rho_1, \ldots, \rho_7$. The MCMC sampler took 4000 burn-in runs of the parameters and 4000 production runs for each of the 100 data sets. A subset of 200 posterior draws was kept from each production run where these were spaced 20 samples apart. This resulted in 20,000 (200×100) posterior draws for each of the six ρ parameters. The medians of the ρ parameter posterior draws are listed in Table 7.13. The upper 5, 10, and 15% quantiles of the 100 medians of the ρ_7 reference distribution were 0.9844, 0.9812, and 0.9781, respectively.

Boxplots of the posterior draws of $\rho_1, \ldots, \rho_6$ are shown in Fig. 7.15 along with horizontal reference lines marking the 5, 10, and 15% quantiles of the benchmark distribution. From most to least important, the procedure identifies *clearance*, *pocket width*, *punch plan view radius*, and *fillet fadius* as active inputs. Neither *pocket length* nor *lock bead distance* greatly influences failure depth, at least over the input ranges that this simulator has been run. ♦

Input	Median
Clearance	0.7765
Fillet radius	0.9464
Punch plan view radius	0.9005
Pocket width	0.8641
Pocket length	1.0000
Lock bead distance	1.0000

Table 7.13 Medians of the 20,000 posterior draws of the correlation parameters $\rho_1, \ldots, \rho_6$

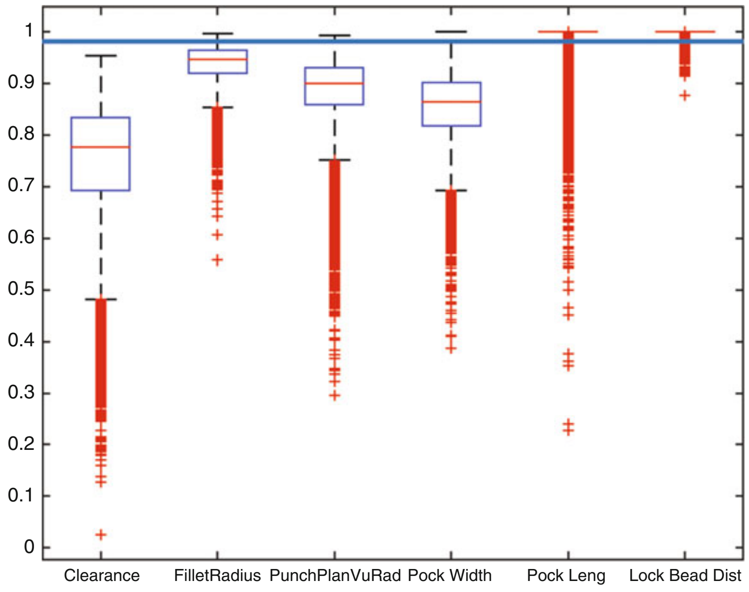


Fig. 7.15 Boxplots of $\rho_1, \ldots, \rho_6$ posterior draws that are combined from 100 failure depth data sets obtained by adding (7.6.5) to the failure depths. The horizontal lines are of the posterior draws of the 5, 10, and 15% upper quantiles of 100 medians of the ρ_7 draws

7.7 Chapter Notes

This chapter is but an introduction to the large and growing literature on sensitivity analysis. Readers wishing a more comprehensive discussion of local and global sensitivity measures can consult the classical resources Saltelli et al. (2000, 2004), among others.

7.7.1 Designing Computer Experiments for Sensitivity Analysis

Saltelli et al. (2000, 2010), Saltelli (2002), Chen et al. (2005), Helton et al. (2006), Morris et al. (2006, 2008), Marrel et al. (2009), and Storlie et al. (2009) and the references therein propose designs for computer experiments and method of moments estimators of the variance terms that comprise sensitivity indices and hence of the sensitivity indices themselves. These methods can be thought of as replacing the assumption that the output can be represented as a draw from a certain process with the additional sampling that is required to provide moment estimators of the required means and variances in the sensitivity indices.

Storlie and Helton (2008) and the references therein describe smoothing and metamodel-based methods. Sobol´ and Kucherenko (2009) provide links between derivative-based measures and Sobol´ indices.

Campbell (2001) and Campbell et al. (2005, 2006) provide definitions of sensitivity indices for multivariate and functional output and methods for estimating these indices.

7.7.2 Orthogonality of Sobol´ Terms

Among other authors, Van Der Vaart (1998) (Sect. 11.4) shows that any component of (7.4.9), say $y_Q(\boldsymbol{X}_Q)$, where $Q \subset \{1, 2, \ldots, d\}$, must have zero mean when integrated with respect to any input X_i, with $i \in Q$, and any pair of terms in (7.4.9) must be *pairwise orthogonal*. We give a proof of these two facts that uses the notation and definitions of Sect. 7.4.

Lemma 7.1. For $Q = \{j_1, \ldots, j_s\} \subseteq \{1, \ldots, d\}$,

$$\int_0^1 y_Q(\boldsymbol{x}_Q)\, dx_{j_k} = 0 \qquad (7.7.1)$$

for any $j_k \in Q$.

Proof: The proof proceeds by induction on the number of elements in Q. When $Q = \{j\}$, say, then from (7.4.1), (7.4.2), and the definition of y_0, (7.7.1) holds for any main effect function $y_j(x_j)$. Suppose that $Q \subseteq \{1, \ldots, d\}$ contains two or more elements, and assume that (7.7.1) holds for all proper subsets $E \subset Q$. Fix $\ell \in Q$, and let $Q \backslash \ell$ denote the set difference of Q and $\{\ell\}$, which is non-empty by definition of Q. Partition the non-empty subsets of Q into the collection $\mathcal{U}_+$ of subsets E that *contain* ℓ, and the collection $\mathcal{U}_-$ of subsets E that *do not contain* ℓ; note that $Q \backslash \ell \in \mathcal{U}_-$. Then by the definition (7.4.12) of $y_Q(\boldsymbol{x}_Q)$,

$$\begin{aligned}\int y_Q(\boldsymbol{x}_Q)\, dx_\ell &= \int \left\{ u_Q(\boldsymbol{x}_Q) - \sum_{E \subset Q} y_E(\boldsymbol{x}_E) - y_0 \right\} dx_\ell \\ &= \int u_Q(\boldsymbol{x}_Q)\, dx_\ell - \sum_{E \in \mathcal{U}_+} \int y_E(\boldsymbol{x}_E)\, dx_\ell \qquad (7.7.2) \\ &\quad - \sum_{E \in \mathcal{U}_-} y_E(\boldsymbol{x}_E) - y_0\,, \end{aligned}$$

where the third and fourth terms use the fact that their integrands do not depend on x_ℓ (because $\ell \notin E$ for $E \in \mathcal{U}_-$).

By definition of $u_Q(\boldsymbol{x}_Q)$, the first term of (7.7.2) is

$$\int u_Q(\boldsymbol{x}_Q)\, dx_\ell = \int \int y(\boldsymbol{x}_Q, \boldsymbol{x}_{-Q})\, d\boldsymbol{x}_{-Q}\, dx_\ell = u_{Q\setminus\ell}(\boldsymbol{x}_{Q\setminus\ell})\,.$$

The second term of (7.7.2) is zero since (7.7.1) holds for all proper subsets of Q by assumption. This gives that (7.7.2) is

$$\int y_Q(\boldsymbol{x}_Q)\, dx_\ell = u_{Q\setminus\ell}(\boldsymbol{x}_{Q\setminus\ell}) - 0 - \left(\sum_{E \in \mathcal{U}_- ; E \neq Q\setminus\ell} y_E(\boldsymbol{x}_E) + y_{Q\setminus\ell}(\boldsymbol{x}_{Q\setminus\ell}) \right) - y_0\,,$$

which is zero by definition of $y_{Q\setminus\ell}(\boldsymbol{x}_{Q\setminus\ell})$. ■

Notice that Lemma 7.1 implies that the mean of each $y_Q(\boldsymbol{X}_Q)$ with respect to $\boldsymbol{X}_Q$ is zero, i.e., $E[y_Q(\boldsymbol{X}_Q)] = 0$ for any $Q \subseteq \{1, \ldots, d\}$. This is a stronger form of centering than that of $u_Q(\boldsymbol{X}_Q)$ by y_0 which also satisfies $E[u_Q(\boldsymbol{x}_Q) - y_0] = 0$ but for which $\int_0^1 (u_Q(\boldsymbol{x}_Q) - y_0)\, dx_{j_k}$ need not be zero for any $j_k \in Q$.

Lemma 7.1 also implies that the orthogonality in (7.4.14) holds. Suppose that $(i_1, \ldots, i_s) \neq (j_1, \ldots, j_t)$; pick any integer k that is in exactly one of $(i_1, \ldots, i_s)$ or $(j_1, \ldots, j_t)$ (there has to be at least one such integer), and integrate

$$\int_0^1 \cdots \int_0^1 y_{i_1,\ldots,i_s}(x_{i_1}, \ldots, x_{i_s}) \times y_{j_1,\ldots,j_t}(x_{j_1}, \ldots, x_{j_t}) \prod_\ell dx_\ell$$

in the order: k and then over $(i_1, \ldots, i_s) \cup (j_1, \ldots, j_t) \setminus k$ (in any order). The inner integral is zero and thus (7.4.14) holds.

7.7.3 Weight Functions $g(\boldsymbol{x})$ with Nonindependent Components

While one can formally define average $y(\boldsymbol{x})$ values with respect to an arbitrary weight function $g(\boldsymbol{x})$ by

$$u_k(x_k) = \int_0^1 \cdots \int_0^1 y(x_1, \ldots, x_d)\, g(\boldsymbol{x}) \prod_{\ell \neq k} dx_\ell$$

and an analog of the Sobol´ decomposition (7.4.9), neither the orthogonality (7.4.14) nor zero mean properties (7.4.13) need hold. This situation is similar to that in ANOVA modeling with “unbalanced” data.

7.7.4 Designs for Estimating Elementary Effects

There have been numerous advances in the design of experiments for estimating elementary effects. Pujol (2009) introduced a *non-collapsing* OAT design, i.e., a design for which projections of the input vectors of the design are not identical. Campolongo et al. (2007) propose a criterion for *spreading the set of complete tours* to more fully explore the input space; they also suggest that using $\overline{|d_j|} = \frac{1}{r}\sum_{i=1}^{r}|d_j(\boldsymbol{x}_i^j)|$ in place of $\overline{d_j}$ and S_j gives equivalent information and a simpler analysis tool.

Campolongo et al. (2011) introduce *radial OAT designs* that spread starting points by using a Sobol´ sequence and allow for differential Δ for each input. Sun et al. (2013) describe an alternative method to Campolongo et al. (2011) for spreading tours widely over the input space (and hence to facilitate more accurate descriptions of the output function); they also describe an experimental design that can be used to provide elementary effects for non-rectangular input regions.

7.7.5 Variable Selection

Other $y(\boldsymbol{x})$ predictors have been used in conjunction with variable selection, for example, blind kriging and the Bayesian composite Gaussian process predictor (Joseph et al. (2008); Davis (2015)). However the papers discussed in Sect. 7.6 make explicit use of reference distributions and how to apply design to enhance variable selection.

An entirely different approach to reducing the dimension of the input space is described in Russi (2010) and Constantine et al. (2014). These authors propose reducing the dimensionality of a problem by identifying the "active subspace" of inputs, i.e., finding a linear transformation $\boldsymbol{T}$ for which the output depends only on $\boldsymbol{T}^\top \boldsymbol{x}$.

7.7.6 Global Sensitivity Indices for Functional Output

Campbell et al. (2005) describe an approach to defining sensitivity indices for *functional output*. In outline, the idea of the method is described in the following paragraphs. The notation used in this section is as follows. The $m \times 1$ multivariate output $\boldsymbol{y}(\boldsymbol{x}) = \left(y^1(\boldsymbol{x}), \ldots, y^m(\boldsymbol{x})\right)^\top$ is observed at inputs $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n_s}$; note that for later ease of notation, the individual outputs are denoted by a superscript rather than a subscript. As usual assume the inputs have a rectangular domain that has been scaled so that $\boldsymbol{x}_i \in [0, 1]^d$, $i = 1, \ldots, n_s$. Finally, it is convenient to let $\boldsymbol{y}_i$ denote the $m \times 1$ vector $\boldsymbol{y}(\boldsymbol{x}_i)$. While the sensitivity analysis tools described below can be applied to the case of low-dimensional multivariate $\boldsymbol{y}(\boldsymbol{x})$ output that result from one or more simulator codes run at $\boldsymbol{x}$, these methods are primarily thought of as tools applied to *functional output* recorded at a discretized set of domain values.

Applying standard multivariate data reduction, let $\boldsymbol{k}_1, \ldots, \boldsymbol{k}_p$ denote an orthogonal basis representation of $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{n_s}$ (assuming $n_s \geq p$); each $\boldsymbol{k}_\ell$ is an $m \times 1$ vector. For smooth functional outputs, one can often find such a basis that captures 99% of the variability in the data but has $p \ll m$ which results in an enormous reduction in the dimension of $\boldsymbol{y}(\boldsymbol{x})$. In practice, a basis $\{\boldsymbol{k}_\ell\}_{\ell=1}^{p}$ can often be obtained from a singular value decomposition of the (row-centered) $m \times n$ matrix

$$\left[\boldsymbol{y}_1\ \boldsymbol{y}_2\ \cdots\ \boldsymbol{y}_{n_s}\right].$$

Let $w_1(\boldsymbol{x}_i), \ldots, w_p(\boldsymbol{x}_i)$ denote the corresponding coefficients in the basis representation of $\boldsymbol{y}_i$ so that the approximation

$$\boldsymbol{y}(\boldsymbol{x}_i) \approx \boldsymbol{k}_1\, w_1(\boldsymbol{x}_i) + \cdots + \boldsymbol{k}_p w_p(\boldsymbol{x}_i), \quad i = 1, \ldots, n_s \tag{7.7.3}$$

has high accuracy. For simplicity of notation, assume that equality holds in (7.7.3), although only approximate equality will hold in $\boldsymbol{y}$-expressions involving the basis vectors.

Applying the Sobol´ decomposition (7.4.9) to the coefficient functions $w_1(\boldsymbol{x}), \ldots, w_p(\boldsymbol{x})$ results in the formula

$$w_\ell(\boldsymbol{x}) = w_0^\ell + \sum_{i=1}^{d} w_i^\ell(x_i) \ + \sum_{1 \leq i < j \leq d} w_{i,j}^\ell(x_i, x_j) + \cdots + w_{1,2,\ldots,d}^\ell(\boldsymbol{x})$$

for $\ell = 1, \ldots, p$, where $\boldsymbol{x} = (x_1, \ldots, x_d)$ and the zero mean and uncorrelated component functions are defined as in (7.4.1). Thus the variance of $w_\ell(\boldsymbol{x})$ can be expressed as the sum of the variances of the $w_Q^\ell(\boldsymbol{x})$ components

$$V^\ell \equiv Var\left[w_\ell(\boldsymbol{X})\right] = \sum_{k=1}^{d} V_k^\ell + \sum_{k<j} V_{kj}^\ell + \cdots + V_{1,\ldots,d}^\ell, \tag{7.7.4}$$

using the variance decomposition (7.4.16) where $V_Q^\ell = Var[w_Q^\ell(\boldsymbol{X}_Q)]$ for non-empty $Q \subset \{1, \ldots, d\}$.

The goal of sensitivity analysis applied to multivariate output data is to divide the total variance of $\boldsymbol{y}(\boldsymbol{x})$,

$$\begin{aligned} V &\equiv \sum_{\ell=1}^{m} \int_{[0,1]^d} \left[y^\ell(\boldsymbol{x}) - \int_{[0,1]^d} y^\ell(\boldsymbol{x})\, d\boldsymbol{x} \right]^2 d\boldsymbol{x} \\ &= \int_{[0,1]^d} \boldsymbol{y}^\top(\boldsymbol{x}) \boldsymbol{y}(\boldsymbol{x})\, d\boldsymbol{x} - \left(\int_{[0,1]^d} \boldsymbol{y}(\boldsymbol{x})\, d\boldsymbol{x} \right)^\top \left(\int_{[0,1]^d} \boldsymbol{y}(\boldsymbol{x})\, d\boldsymbol{x} \right), \end{aligned}$$

into constituents that are attributable to changes in one or more sets of inputs. This is done component by component of $\boldsymbol{y}(\boldsymbol{x})$ but results in simple algebraic expressions when stated in terms of the basis vectors and coefficients of (7.7.3).

To illustrate, the vector of overall means of $\boldsymbol{y}(\boldsymbol{x})$ is the $m \times 1$

$$\boldsymbol{y}_0 = \int_{[0,1]^d} \boldsymbol{y}(\boldsymbol{x})\, d\boldsymbol{x}.$$

For $k = 1, \ldots, d$, the k^{th} main effect of $\boldsymbol{y}(\boldsymbol{x})$ is the vector of k^{th} main effects of the $\boldsymbol{y}(\boldsymbol{x})$ components

$$\boldsymbol{y}_k(x_k) = \int \boldsymbol{y}(\boldsymbol{x}) \prod_{q \neq k} dx_q - \boldsymbol{y}_0\,.$$

Similar expressions hold for higher-order joint effect functions associated with $\boldsymbol{y}(\boldsymbol{x})$.

Beginning with V, main effect and higher-order effect components of $\boldsymbol{y}(\cdot)$ are expressed using (7.7.3). First, it is straightforward to show that substituting (7.7.3) for $\boldsymbol{y}(\boldsymbol{x})$ and using (7.7.4) gives

$$V = \sum_{\ell=1}^{p} \lambda_\ell V^\ell$$

where $\lambda_\ell \equiv \boldsymbol{k}_\ell^\top \boldsymbol{k}_\ell$. The overall mean is

$$\boldsymbol{y}_0 = \boldsymbol{k}_1\, w_0^1 + \cdots + \boldsymbol{k}_p w_0^p\,.$$

The vector of k^{th} main effects of $\boldsymbol{y}(\boldsymbol{x})$, $\boldsymbol{y}_k(x_k)$ is

$$\boldsymbol{y}_k(x_k) = \boldsymbol{k}_1\, w_k^1(x_k) + \cdots + \boldsymbol{k}_p w_k^p(x_k)\,.$$

The $(k, j)^{th}$ joint effect *vector* of $\boldsymbol{y}(\boldsymbol{x})$ is

$$\boldsymbol{y}_{k,j}(x_k, x_j) = \boldsymbol{k}_1\, w_{k,j}^1(x_k, x_j) + \cdots + \boldsymbol{k}_p w_{k,j}^p(x_k, x_j)\,,$$

for $1 \le k < j \le d$. Similar expressions hold for higher-order effect functions.

The k^{th} main effect variance component is the sum of the variances of the components of the main effect vector $\boldsymbol{y}_k(x_k)$ which can be expressed as

$$V_k = \sum_{\ell=1}^{p} \lambda_\ell V_k^\ell, \quad k = 1, \ldots, d.$$

The $(k, j)^{th}$ joint effect variance component is

$$V_{kj} = \sum_{\ell=1}^{p} \lambda_\ell V_{kj}^\ell, \quad 1 \le k < j \le d\,.$$

Similar expressions hold for higher-order effects.

Finally, functional sensitivity indices are defined for the k^{th} input by

$$S_k = V_k / V, \quad k = 1, \ldots, d\,,$$

and the $(k, j)^{th}$ pair of inputs by

$$S_{kj} = V_{kj}/V, \quad 1 \le k < j \le d\,.$$

Higher-order sensitivity indices can be defined analogously.

7.7.7 Software

The commercial software JMP produces plug-in estimated main effect plots for GP models with Gaussian or cubic correlation functions. It provides plug-in estimates of the main effect (ME), total effect (TE), and pairwise joint effect global sensitivity indices.

Based on a constant mean GP with Gaussian correlation function, the program GPMSA (http://go.osu.edu/GPMSA) produces (fully) Bayesian estimates of the effect functions of any order as well as estimates of the ME and TE global sensitivity indices.

Chapter 8
Calibration

8.1 Introduction

Ideally, every computer simulator should be *calibrated* using observations from the physical system that is modeled by the simulator. Roughly, calibration uses data from dual simulator and physical system platforms to *estimate*, with uncertainty, the *unknown values of the calibration inputs* that govern the physical system (and which can be set in the simulator). Another objective of calibration is to predict the *mean of the physical system observations* based on the data from both platforms.

Unfortunately, observations on the physical system need not be always available. In some cases this is due to the cost of conducting the desired physical system experiment and in others to ethical considerations. Even when experiments using the full physical system are not possible, sometimes subsystem experiments can be conducted. In yet other cases, experiments can be run using an approximation to the ideal physical system experiment; an example of the latter occurs in biomechanical applications where cadaver joints are used in place of the joints of living subjects.

This chapter describes a method for calibrating a computer simulator based on physical system observations. To begin this discussion, two examples from Sect. 1.2 will be reviewed that contain the elements present in calibration problems. Example 1.4 described the optimization of an injection molding process for mass-producing precision plastic components such as plastic camera bodies as discussed by Villarreal-Marroquín et al. (2017). Their goal was to minimize shrinkage in a range of plastic test components by setting four manufacturing process control variables: the melting time of the plastic pellets; the time to pack the mold; the pressure used during packing; and the time that a part is cooled in the mold. Data were available from both a physical system experiment using an injection molding machine and from a computer experiment that used a commercial code (Moldex3D) which simulates the molding process. In addition to the manufacturing process control variables, the simulator required the specification of three heat transfer coefficients (HTCs) of the mold during flow, packing, and cooling; these values were not completely known, but expert opinion of their value was available. The second illustration of the dual experimental platform elements required by calibration methodology is given in Example 1.7. This example described the behavior of materials in a high strain rate environment where the output is *functional*. Specifically the output was the velocity of a free surface at 136 time points. Because the physical experiment is run for a *single condition* that is mimicked by the simulator, one should think of the unstated control input(s) being fixed at one value. The simulator had 10 calibration parameters corresponding to unknown model and material property assumptions about the flyer plate (see Table 1.4). The calibration parameters were varied over 128 configurations of model/material values in the simulator runs.

The features from both examples that are common to all calibration problems are:

- a single set of control inputs that can be modified or observed in both the physical system and its simulator;
- an additional set of calibration inputs to the simulator that specify the characteristics of the model implemented or conditions under which the physical system operates;
- a real-valued, multivariate, or functional output both computed by the simulator and observed from the physical system.

A more detailed description of calibration data will be given in Sects. 8.2 and 8.4.

One final example that allows calibration will be presented to suggest the wide range of applications of this methodology. Bayarri et al. (2007) describe data used to determine the yield stress and other material properties of a spot weld. Spot welds are formed between two pieces of sheet metal by tightly compressing the sheets between a pair of electrodes and applying a high voltage so as to melt the metal at the compressed point forming a "nugget" at the weld site. The important factors that determine the size of the nugget produced by this process include the load/pressure at the weld site, the thickness of the metal pieces, the voltage applied, and the electrical resistance between the electrodes. Both physical system data from a series of manufactured spot welds and from a computer simulator of this process were available to construct a predictor of the mean size of the nugget produced in the manufacturing process.

Before describing calibration methodology, two additional terms, *verification* and *validation*, will be introduced that can be confused easily with each other and with the process of calibration. Both terms are used in the setting where there is a computer simulator that is meant to implement a mathematical model of a given physical system.

Validation is the process of insuring that a given mathematical model correctly represents a specified physical system. In engineering, the terminology "validation" often means the process of selecting a set of "calibration" inputs at which to run a simulator so that the simulator most accurately mimics the modeled physical system for a range of the control inputs.

Verification is the process of determining that a given computer software correctly solves a specified mathematical model over a stated range of inputs. For example, the `MASA` (Manufactured Analytical Solution Abstraction) software that is distributed at http://pecos.ices.utexas.edu/software provides software verification of partial differential equation solvers in multiple dimensions.

Both verification and validation of a simulator are essential first steps in its use. Many engineering, scientific, and other sectors have developed procedures to ensure that the codes used in their arenas have been thus vetted. Two examples are:

- The mission of the Center for Predictive Engineering and Computational Science at the University of Texas, Austin (http://pecos.ices.utexas.edu), is to develop and apply tools and techniques for making reliable computational predictions of complex systems, mainly in the physical sciences and engineering.
- The US FDA is in the process of establishing guidelines for the verification and validation of the computational models of medical devices (https://cstools.asme.org/csconnect/CommitteePages.cfm?Committee=100108782).

Sections 8.2 and 8.3 introduce, for real-valued outputs, the Kennedy and O'Hagan (2001) model and present cautions about the use of this method, especially for making inference about calibration parameters. Sections 8.4 and 8.5 extend the calibration methodology to functional output. Section 8.6 provides supplementary information about calibration methodology, including references to software implementations.

8.2 The Kennedy and O'Hagan Calibration Model

8.2.1 Introduction

This section describes the Kennedy and O'Hagan (2001) calibration model (the "KOH" model) and several approaches that use it for statistical inference. Then Sect. 8.3 will adopt one of these methods, a fully Bayesian approach of the KOH model as described by Higdon et al. (2004) and Higdon et al. (2008). A simple analytic example will be presented that illustrates the methods and provides a caution about how the model can be misused. Finally a real-data example will be given which illustrates calibration methodology.

8.2.2 The KOH Model

The KOH model assumes that the output from both a physical experiment and a computer simulator experiment of that system is available. Each input to these experiments is either a *control input* or a *calibration input*. Control inputs can be set by the researcher; all inputs to a (completely randomized) physical experiment are control variables. The model assumes that all control inputs to the physical experiment can also be set in simulator runs. In addition, the simulator requires specification of physical constants such as growth rates or material properties that are collectively termed calibration parameters and are not known exactly. In the examples of Chap. 1, the heat transfer coefficients in the injection molding model of Example 1.4 and the elastic moduli of the component materials in the flyer plate experiment of Example 1.7 are calibration parameters.

More formally, let n denote the number of observations that are taken from a physical system when the control variables have been set to $\boldsymbol{x}_1^p, \ldots, \boldsymbol{x}_n^p$, respectively. Assume that every $\boldsymbol{x}_i^p = (x_{i,1}^p, \ldots, x_{i,d}^p)^\top$ consists of d input variables. Let $y^p(\boldsymbol{x}_1^p), \ldots, y^p(\boldsymbol{x}_n^p)$ denote the resulting physical system observations (the $y^p(\boldsymbol{x}_i^p)$ are often termed *field data*). Similarly let m denote the number of simulator runs made. Let the input settings for the m runs be denoted $(\boldsymbol{x}_1^s, \boldsymbol{t}_1), \ldots, (\boldsymbol{x}_m^s, \boldsymbol{t}_m)$ where $\boldsymbol{x}_i^s = (x_{i,1}^s, \ldots, x_{i,d}^s)^\top$ is the control portion of the i^{th} simulator input and $\boldsymbol{t}_i = (t_{i,1}, \ldots, t_{i,q})^\top$ is the calibration portion of this input. The control inputs in $\boldsymbol{x}^p$ and $\boldsymbol{x}^s$ represent the *same variables*. The simulator outputs will be denoted $y^s(\boldsymbol{x}_1^s, \boldsymbol{t}_1), \ldots, y^s(\boldsymbol{x}_m^s, \boldsymbol{t}_m)$. The notation used by the KOH model is summarized in Table 8.1.

Symbol	Quantity denoted
n	Number of experimental runs using the physical system
m	Number of simulator runs
d	Number of control inputs
$\boldsymbol{x}$	$d \times 1$ vector of control inputs, either to a physical system or a simulator
q	Number of calibration parameters (inputs)
$\boldsymbol{t}$	$q \times 1$ vector of calibration parameters
$y^p(\boldsymbol{x})$	Physical system observation conducted at (control) input $\boldsymbol{x}$
$y^s(\boldsymbol{x}, \boldsymbol{t})$	Output from a simulator run at control input $\boldsymbol{x}$ and calibration parameter $\boldsymbol{t}$

Table 8.1 Notation used to describe the data employed in calibration

The KOH model has two important features. First, it allows for imperfect simulators, i.e., when using the same set of control inputs, the KOH model permits the simulator output to deviate from the mean of the physical experiment. Further, such deviation is permitted even if the simulator is run using the "true" values of the calibration inputs. Second, it assumes that the simulator is sufficiently accurate and that subject-matter experts can place a prior on the calibration parameters that is consistent with the values used by the physical system. In other words, the model is Bayesian with a prior assumed to be based on subject-matter expert opinion of the unknown model parameters and also on the unknown calibration parameters. Example 8.1 presents a simple example of the model elements.

KOH uses a two-stage hierarchical Bayesian model to link the simulator runs and the physical system observations. Similar to the two-stage hierarchical model described in Chap. 4, the probabilistic mechanism used in this chapter can be viewed as selecting a set of model parameters from a specified distribution and, given these parameters, choosing $y^s(\boldsymbol{x}^s, \boldsymbol{t})$ and $y^p(\boldsymbol{x}^p)$ as the realizations of the parameterized stochastic model.

This chapter assumes the outputs $y^s(\boldsymbol{x}_1^s, \boldsymbol{t}_1), \ldots, y^s(\boldsymbol{x}_m^s, \boldsymbol{t}_m)$ of the m simulator runs can, conditionally, be viewed as values of a function $y^s(\boldsymbol{x}, \boldsymbol{t})$ which has been drawn from a stationary Gaussian process (GP) $Y^s(\boldsymbol{x}, \boldsymbol{t})$. This chapter assumes that the GP has mean β_0, process precision λ_s, and $m \times m$ correlation matrix $\boldsymbol{R}_s$ whose $(i, j)^{th}$ element is $R_s\left((\boldsymbol{x}_i^s, \boldsymbol{t}_i), (\boldsymbol{x}_j^s, \boldsymbol{t}_j)\right)$ where

$$R_s\left((\boldsymbol{x}_1, \boldsymbol{t}_1), (\boldsymbol{x}_2, \boldsymbol{t}_2)\right) = \prod_{k=1}^{d} (\rho_k^x)^{(x_{1,k}-x_{2,k})^2} \prod_{\ell=1}^{q} (\rho_\ell^t)^{(t_{1,\ell}-t_{2,\ell})^2} \tag{8.2.1}$$

(see Sect. 2.2.2). The vectors $\boldsymbol{\rho}^x = (\rho_1^x, \ldots, \rho_d^x)$ and $\boldsymbol{\rho}^t = (\rho_1^t, \ldots, \rho_q^t)$ are referred to as the correlation parameters of this model. In sum, $Y^s(\boldsymbol{x}, \boldsymbol{t})$ is conditionally a GP given parameters $(\beta_0, \lambda_s, \boldsymbol{\rho}^x, \boldsymbol{\rho}^t)$ which will be denoted informally by

$$\left[Y^s(\boldsymbol{x}^s, \boldsymbol{t}) \,\middle|\, \beta_0, \lambda_s, \boldsymbol{\rho}^x, \boldsymbol{\rho}^t\right] \sim GP\left(\beta_0, \lambda_s, (\boldsymbol{\rho}^x, \boldsymbol{\rho}^t)\right) \tag{8.2.2}$$

assuming the correlation $R_s(\cdot, \cdot)$ in (8.2.1).

In some cases it can be useful to allow the generalization:

$$E\left[Y^s(\boldsymbol{x}^s, \boldsymbol{t}) \,\middle|\, \boldsymbol{\beta}, \lambda_s, \boldsymbol{\rho}^x, \boldsymbol{\rho}^t\right] = \sum_{j=1}^{p} f_j(\boldsymbol{x}^s, \boldsymbol{t})\, \beta_j = \boldsymbol{f}^\top(\boldsymbol{x}^s, \boldsymbol{t})\, \boldsymbol{\beta} \tag{8.2.3}$$

of the constant mean model (8.2.2), where the regression functions $f_j(\boldsymbol{x}, \boldsymbol{t})$, $j = 1, \ldots, p$, are known and $\boldsymbol{\beta} \in \mathbb{R}^p$ is unknown. The extension (8.2.3) provides a non-stationary GP whose analysis requires more notation than that of the constant mean GP but no additional theoretical development.

Despite the availability of this generalization, in practice $Y^s(\boldsymbol{x}^s, \boldsymbol{t})$ is often assumed to have *mean zero* after the simulator data is standardized to have sample mean zero. Similarly, the Gaussian correlation function (8.2.1) can be replaced by one of its equivalent parameterizations, as listed in (2.2.9). However, not all parametric forms of the correlation function are equally interpretable and hence are not equally straightforward to specify priors.

The observations $y^p(\boldsymbol{x}_1^p), \ldots, y^p(\boldsymbol{x}_n^p)$ are assumed to conditionally follow the regression model:

$$Y^p(\boldsymbol{x}_i^p) = \mu(\boldsymbol{x}_i^p) + \epsilon(\boldsymbol{x}_i^p), \quad i = 1, \ldots, n\,, \tag{8.2.4}$$

given $\mu(\boldsymbol{x}^p)$ and $\lambda_\epsilon > 0$. Here $\mu(\boldsymbol{x}^p)$ denotes the mean response of the physical system when the control variables are set at $\boldsymbol{x}^p$ and $\epsilon(\boldsymbol{x}_1^p), \ldots, \epsilon(\boldsymbol{x}_n^p)$ are independent $N(0, \sigma_\epsilon^2)$ measurement errors with unknown variance σ_ϵ^2 (precision $\lambda_\epsilon \equiv 1/\sigma_\epsilon^2$). The enhancement that (8.2.4) provides over the usual regression model is that the form of $\mu(\boldsymbol{x}^p)$ is nonparametric and unspecified.

The KOH model gives $\mu(\boldsymbol{x}^p)$ a GP prior which is defined indirectly in terms of a *bias* (or discrepancy) function. To define the bias, let $\boldsymbol{\theta}$ denote the "true" value of the calibration parameter in the physical system. In this case, the Bayesian model assumes that our knowledge about $\boldsymbol{\theta}$ can be quantified by a prior distribution determined by subject-matter experts. While there may be a single true value, in some applications, the true value of the calibration parameter can vary from one physical experiment to the next. For example, in determining the distribution of stresses at the bone prosthesis of a joint replacement, it is reasonable to assume that, across a target *patient population*, there is a *distribution* of bone elasticities (hence stresses) rather than a single bone elasticity for all patients. In such a case, it is reasonable to regard the *mean* of this distribution as $\boldsymbol{\theta}$. In either of the two settings described above, the notation $\boldsymbol{\Theta}$ is used to denote a random vector endowed with either the Bayesian prior for $\boldsymbol{\theta}$ or the $\boldsymbol{\theta}$ target distribution. The bias of the simulator is defined to be

$$\delta(\boldsymbol{x}^p) \equiv \mu(\boldsymbol{x}^p) - y^s(\boldsymbol{x}^p, \boldsymbol{\theta}), \tag{8.2.5}$$

i.e., the error in the simulator when run at the true value of the calibration parameter.

Using the same hierarchical modeling approach as for the simulator model, let $\boldsymbol{\Omega} = (\boldsymbol{\beta}, \boldsymbol{\lambda}, \boldsymbol{\rho}, \boldsymbol{\Theta})$, denote the set of all parameters including $\lambda_\delta > 0$ and $\boldsymbol{\rho}^\delta = (\rho_1^\delta, \ldots, \rho_d^\delta) \in (0,1]^d$. Here $\boldsymbol{\lambda}$ denotes $(\lambda_s, \lambda_\epsilon, \lambda_\delta)^\top$, and $\boldsymbol{\rho}$ denotes $(\boldsymbol{\rho}^x, \boldsymbol{\rho}^t, \boldsymbol{\rho}^\delta)$. Conditionally given $\boldsymbol{\Omega}$, the bias function $\delta(\boldsymbol{x})$ is assumed to be a realization from

$$[\Delta(\boldsymbol{x}) \mid \boldsymbol{\Omega}\,] \sim GP\left(0, \lambda_\delta, \boldsymbol{\rho}^\delta\right) ;$$

$\Delta(\boldsymbol{x})$ is assumed to have (conditional) $n{\times}n$ correlation matrix $\boldsymbol{R}_\delta$ with $(i,j)^{th}$ element:

$$R_\delta\left(\boldsymbol{x}_i^p, \boldsymbol{x}_j^p\right) = \prod_{\ell=1}^{d} (\rho_\ell^\delta)^{\left(x_{i,\ell}^p - x_{j,\ell}^p\right)^2} . \tag{8.2.6}$$

In addition, $\Delta(\boldsymbol{x})$ is assumed conditionally independent of $Y^s(\boldsymbol{x}, \boldsymbol{\Theta})$ and the measurement error $\epsilon(\boldsymbol{x})$. Thus given $\boldsymbol{\Omega}$, $\mu(\boldsymbol{x}^p) = \delta(\boldsymbol{x}^p) + y^s(\boldsymbol{x}^p, \boldsymbol{\theta})$ is modeled as a realization from the process

$$M(\boldsymbol{x}^p) = \Delta(\boldsymbol{x}^p) + Y^s(\boldsymbol{x}^p, \boldsymbol{\Theta})$$

which is a sum of independent GPs and hence a GP. Similarly given $\boldsymbol{\Omega}$

$$Y^p(\boldsymbol{x}^p) = M(\boldsymbol{x}^p) + \epsilon(\boldsymbol{x}^p) = \Delta(\boldsymbol{x}^p) + Y^s(\boldsymbol{x}^p, \boldsymbol{\Theta}) + \epsilon(\boldsymbol{x}^p) \tag{8.2.7}$$

is the sum of a GP and an independent measurement error.

8.2.2.1 Alternative Views of Calibration Parameters

Before continuing with illustrations of the Bayesian calibration methodology in the remainder of this chapter, the reader should observe the following caveats regarding the definition of $\boldsymbol{\theta}$.

In some applications, simplified theoretical physics or biology models are used to construct the simulator. Thus even if expert knowledge is available about $\boldsymbol{\theta}$ in the physical system, that information may not accurately describe the functioning of these parameters in the simplified simulator.

In other applications, simulation models are mechanistic with unknown parameters representing material properties, say, of the model components. As an example, consider a deterministic biomechanical model of a joint that substitutes springs for ligaments and uses inputs that specify the stiffnesses and other characteristics for each spring. While such inputs are tempting to use as calibration parameters, it would be difficult or impossible to state prior distributions for their values, although it might be feasible to specify the range of each. In other cases, numerical tuning parameters might be considered as calibration parameters, but again it would be difficult to specify priors for such quantities. In such settings it makes sense to define target values of $\boldsymbol{t}$ to be $\boldsymbol{\theta}$ values that minimize some metric that measures the difference between $y^s(\boldsymbol{x}, \boldsymbol{\theta})$ and either $\mu(\boldsymbol{x})$ or $y^p(\boldsymbol{x})$. Termed *engineering validation* by some, an early statistical paper that implicitly adopts this formulation is Cox et al. (2001). More recently Tuo and Wu (2016) and Plumlee (2017) studied this idea in much greater detail. Han et al. (2009b) proposed a method for settings in which some elements of $\boldsymbol{t}$ have priors provided by subject-matter experts, while others are selected by a suitable metric between the quantity to be predicted and a given predictor.

Example 8.1 (A Simple Analytic Example). This toy example illustrates an "alternative" calibration situation as discussed in the previous paragraph. There are two objectives. It will demonstrate the formal Bayesian calibration procedure that is introduced in the following section. The example will also illustrate the influence of the prior parameters of the bias function and show the consequences of using the Bayesian methodology with inaccurate prior information.

This example uses a single control variable which, to simplify notation, is denoted by x throughout rather than x^p or x^s. It uses a real-valued calibration parameter which is denoted t.

Suppose $n = 5$; observations $y^p(x)$ are taken from

$$Y^p(x) = \mu(x) + \epsilon(x), \qquad x \in [-5, 5],$$

at the x values listed in the first column of Table 8.2 where the quadratic expression $\mu(x) = 0.1x^2 - x + 0.4$ is the mean output and $\epsilon(x)$ are measurement errors which, for different x, are independent and identically (i.i.d.) normally distributed with *mean zero* and *variance* 0.8^2. In symbols, the measurement errors are i.i.d. $N(0, 0.8^2)$. As always, stating the variance of the measurement error is equivalent to providing the precision $\lambda_\epsilon = 1/(0.8^2)$.

The data in Table 8.2 are taken to be the "physical system" data for this example; they are plotted in Fig. 8.1. There are "substantial" deviations from the mean $\mu(x)$ in the physical data.

x	$y^p(x)$
−5.0	+7.079
−2.5	+4.538
0	−1.539
2.5	−2.096
5.0	−0.824

Table 8.2 Five observations from the model $y^p(x) = 0.1x^2 - x + 0.4 + N(0, 0.8^2)$

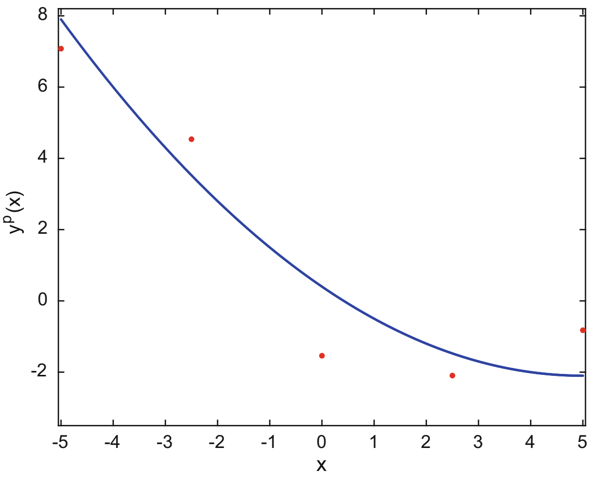


Fig. 8.1 $\mu(x) = 0.1x^2 - x + 0.4$ (blue); the $n = 5$ physical system $y^p(x)$ from Table 8.2 (red)

In addition, $m = 15$ runs are made using the "simulator"

$$y^s(x,t) = t - 1.3 * x\,.$$

For appropriate intercept t, the straight line $y^s(x,t)$ can roughly mimic the quadratic $\mu(x)$ over the control input range $x \in [-5, 5]$ where $t \in [-5, 5]$ is taken to be the calibration parameter. The (x,t) pairs are selected to form a 15 run maximin LHD (see Fig. 8.2). The wide range of t values causes the simulator outputs to be widely dispersed.

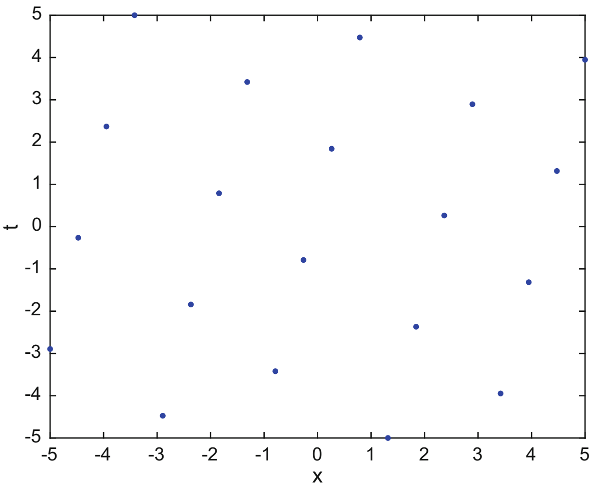


Fig. 8.2 The $m = 15$ simulator inputs (x,t) at which $y^s(x,t)$ has been calculated

The statistical goals are to predict the mean of the physical system, to quantify the posterior uncertainty of the predicted mean, and to perform Bayesian inference about the calibration parameter. ♦

Section 8.3 will describe the Bayesian methodology that is appropriate for cases where all the calibration parameters can be given priors based on expert subject-matter opinion. It will illustrate the methodology and inferences using the simple Example 8.1 where answers can be visualized and then for Example 1.4.

8.3 Calibration with Univariate Data

Let $\boldsymbol{y}^p = (y^p(\boldsymbol{x}_1^p), \ldots, y^p(\boldsymbol{x}_n^p))^\top$ denote the $n \times 1$ vector of physical system observations available for calibration and $\boldsymbol{y}^s = (y^s(\boldsymbol{x}_1^s, \boldsymbol{t}_1), \ldots, y^s(\boldsymbol{x}_m^s, \boldsymbol{t}_m))^\top$ the $m \times 1$ vector of associated computer simulator output. Then $\mathcal{Y} = ((\boldsymbol{y}^p)^\top, (\boldsymbol{y}^s)^\top)^\top$ is the combined $(m + n) \times 1$ vector of physical system and simulator output.

Assuming (8.2.2) and (8.2.7), the kernel of the conditional log likelihood of $\mathcal{Y}$ is

$$\ell n\left([\mathcal{Y} \mid \boldsymbol{\Omega}]\right) = -\frac{1}{2}\, \ell n \det(\boldsymbol{\Sigma}_{\mathcal{Y}}) - \frac{1}{2}\left\{(\mathcal{Y} - \beta_0 \mathbf{1}_{m+n})^\top \boldsymbol{\Sigma}_{\mathcal{Y}}^{-1} (\mathcal{Y} - \beta_0 \mathbf{1}_{m+n})\right\}, \quad (8.3.1)$$

where $\mathbf{1}_{m+n}$ is the $(m + n) \times 1$ vector of ones and

$$\boldsymbol{\Sigma}_{\mathcal{Y}} = \lambda_s^{-1} \begin{bmatrix} \boldsymbol{R}_s^{pp} & \boldsymbol{R}_s^{ps} \\ \boldsymbol{R}_s^{sp} & \boldsymbol{R}_s^{ss} \end{bmatrix} + \lambda_\delta^{-1} \begin{bmatrix} \boldsymbol{R}_\delta^{pp} & \mathbf{0}_{n,m} \\ \mathbf{0}_{m,n} & \mathbf{0}_{m,m} \end{bmatrix} + \lambda_\epsilon^{-1} \begin{bmatrix} \boldsymbol{I}_n & \mathbf{0}_{n,m} \\ \mathbf{0}_{m,n} & \mathbf{0}_{m,m} \end{bmatrix}. \quad (8.3.2)$$

In (8.3.2), $\mathbf{0}_{r,c}$ is the $r \times c$ matrix of zeros, and the remaining components are as follows. The matrices $\boldsymbol{R}_s^{pp}$ and $\boldsymbol{R}_\delta^{pp}$ are each $n \times n$; their entries are the correlations among the $Y^s(\boldsymbol{x}^p, \boldsymbol{\Theta})$ and $\Delta(\boldsymbol{x}^p)$ terms, respectively; specifically, $\boldsymbol{R}_s^{pp}$ has $(i, j)^{th}$ element $R_s\left((\boldsymbol{x}_i^p, \boldsymbol{\Theta}), (\boldsymbol{x}_j^p, \boldsymbol{\Theta})\right)$ while $\boldsymbol{R}_\delta^{pp}$ has $(i, j)^{th}$ element $R_\delta(\boldsymbol{x}_i^p, \boldsymbol{x}_j^p)$, $i, j = 1, \ldots, n$. The $n \times m$ matrix $\boldsymbol{R}_s^{ps} = \left(R_s\left((\boldsymbol{x}_i^p, \boldsymbol{\Theta}), (\boldsymbol{x}_j^s, \boldsymbol{t}_j)\right)\right)$ corresponds to cross products of $Y^s(\boldsymbol{x}_i^p, \boldsymbol{\Theta})$ with $Y^s(\boldsymbol{x}_j^s, \boldsymbol{t}_j)$, $i = 1, \ldots, n$ and $j = 1, \ldots, m$; $\boldsymbol{R}_s^{sp} = (\boldsymbol{R}_s^{ps})^\top$; $\boldsymbol{R}_s^{ss} = \left(R_s\left((\boldsymbol{x}_i^s, \boldsymbol{t}_i), (\boldsymbol{x}_j^s, \boldsymbol{t}_j)\right)\right)$ is the $m \times m$ correlation matrix among the simulator data $Y^s(\boldsymbol{x}_j^s, \boldsymbol{t}_j)$, $j = 1, \ldots, m$. The form of (8.3.1) shows that $[\mathcal{Y} \mid \boldsymbol{\Omega}]$ is multivariate normally distributed with each component having mean β_0 and joint covariance $\boldsymbol{\Sigma}_{\mathcal{Y}}$.

Given the prior $[\boldsymbol{\Omega}]$ for $\boldsymbol{\Omega}$, the expression (8.3.1) for the kernel of $[\mathcal{Y} \mid \boldsymbol{\Omega}]$ immediately gives

$$[\boldsymbol{\Omega} \mid \mathcal{Y}] \propto [\mathcal{Y} \mid \boldsymbol{\Omega}] \times [\boldsymbol{\Omega}] \quad (8.3.3)$$

for the kernel of the posterior distribution of $\boldsymbol{\Omega}$ given $\mathcal{Y}$. In many cases, (8.3.3) will not correspond to any well-known distribution, but MCMC methods can be used to draw samples from it (see Appendix D).

8.3.1 Bayesian Inference for the Calibration Parameter $\boldsymbol{\Theta}$

Samples from the (marginal) posterior $[\boldsymbol{\Theta} \mid \mathcal{Y}]$ of $\boldsymbol{\Theta}$ can be obtained from samples drawn from $[\boldsymbol{\Omega} \mid \mathcal{Y}]$; these draws provide a data-based update to the $\boldsymbol{\Theta}$ prior. The expectation $E[\boldsymbol{\Theta} \mid \mathcal{Y}]$ of the posterior provides a point prediction of the calibration parameter. If $\boldsymbol{\Theta}$ is a real-valued parameter, say Θ, then either forming a highest posterior density set for Θ *or* calculating upper and lower $\alpha/2$ quantiles of the $[\Theta \mid \mathcal{Y}]$ distribution can be used to quantify the Bayesian uncertainty about Θ as a $100 \times (1 - \alpha)\%$ credible set. If $\boldsymbol{\Theta}$ is a vector with q components, then forming a joint credible set can be difficult, although a highest posterior density region can, in principle, be generated. More simply, constructing $100 \times (1 - \alpha/q)\%$ individual credible intervals for each component of $\boldsymbol{\Theta}$ yields a joint $100 \times (1 - \alpha)\%$ credible hyper-rectangle for the vector $\boldsymbol{\Theta}$.

8.3.2 Bayesian Inference for the Mean Response $\mu(\boldsymbol{x})$ of the Physical System

Let $\boldsymbol{x}$ denote a vector of control variables. Recall from (8.2.7) that

$$Y^p(\boldsymbol{x}) = M(\boldsymbol{x}) + \epsilon(\boldsymbol{x}),$$

where $M(\boldsymbol{x}) = \Delta(\boldsymbol{x}) + Y^s(\boldsymbol{x}, \boldsymbol{\Theta})$. Bayesian inference for $\mu(\boldsymbol{x})$ is derived from the posterior predictive distribution $[M(\boldsymbol{x}) \mid \mathcal{Y}]$. The usual Bayesian predictor of $\mu(\boldsymbol{x})$ is the mean of the posterior distribution, i.e.,

$$\widehat{\mu}(\boldsymbol{x}) = E\left[M(\boldsymbol{x}) \mid \mathcal{Y}\right] = E\left[\Delta(\boldsymbol{x}) + Y^s(\boldsymbol{x}, \boldsymbol{\Theta}) \mid \mathcal{Y}\right] . \tag{8.3.4}$$

The conditional expectation is over the joint distribution of the unseen values of $Y^s(\cdot, \cdot)$ and $\Delta(\cdot)$ as well as over the unknown $\boldsymbol{\Omega}$. Uncertainty bounds for prediction of $\mu(\boldsymbol{x})$ can be obtained from the quantiles of the draws $[M(\boldsymbol{x}) \mid \mathcal{Y}]$. These bounds describe the uncertainty due to not knowing $\delta(\boldsymbol{x})$, nor the form of the simulator $y^s(\boldsymbol{x}, \boldsymbol{t})$, nor the measurement error.

An MCMC method for computing (8.3.4) based on having available draws from $[\boldsymbol{\Omega} \mid \mathcal{Y}]$ is

$$\begin{aligned}
\widehat{\mu}(\boldsymbol{x}) &= E\left[M(\boldsymbol{x}) \mid \mathcal{Y}\right] \\
&= E\left[E\left[M(\boldsymbol{x}) \mid \mathcal{Y}, \boldsymbol{\Omega}\right]\right] \\
&= \int E\left[M(\boldsymbol{x}) \mid \mathcal{Y}, \boldsymbol{\Omega}\right] \, [\boldsymbol{\Omega} \mid \mathcal{Y}] \; d\boldsymbol{\Omega} \\
&\approx \frac{1}{N_{mcmc}} \sum_{q=1}^{N_{mcmc}} E\left[M(\boldsymbol{x}) \mid \mathcal{Y}, \boldsymbol{\Omega}^q\right] ,
\end{aligned} \tag{8.3.5}$$

where $\boldsymbol{\Omega}^q = (\beta_0^q, \boldsymbol{\lambda}^q, \boldsymbol{\rho}^q, \boldsymbol{\Theta}^q)$, $q = 1, \ldots, N_{mcmc}$, are independent draws from $[\boldsymbol{\Omega} \mid \mathcal{Y}]$. A formula to evaluate the mean of $M(\boldsymbol{x})$ given $(\mathcal{Y}, \boldsymbol{\Omega})$ is straightforward to state upon recognizing that

$$[M(\boldsymbol{x}), \mathcal{Y} \mid \boldsymbol{\Omega}]$$

has a multivariate normal distribution because $M(\boldsymbol{x}) = \Delta(\boldsymbol{x}) + Y^s(\boldsymbol{x}, \boldsymbol{\Theta})$ and all processes are Gaussian and independent. Hence Lemma B.2 states that the distribution of $M(\boldsymbol{x})$ given $(\mathcal{Y}, \boldsymbol{\Omega})$ is univariate normal with conditional mean and variance

$$E\left[M(\boldsymbol{x}) \mid \mathcal{Y}, \boldsymbol{\Omega}\right] = \beta_0 + \boldsymbol{\Sigma}_{\boldsymbol{x},\mathcal{Y}}^\top \boldsymbol{\Sigma}_{\mathcal{Y}}^{-1} (\mathcal{Y} - \beta_0 \mathbf{1}_{n+m}),$$

where $\boldsymbol{\Sigma}_{\boldsymbol{x},\mathcal{Y}}$ is the $(n+m) \times 1$ vector

$$\lambda_s^{-1} \begin{bmatrix} \boldsymbol{R}_s^{\boldsymbol{x}p} \\ \boldsymbol{R}_s^{\boldsymbol{x}s} \end{bmatrix} + \lambda_\delta^{-1} \begin{bmatrix} \boldsymbol{R}_\delta^{\boldsymbol{x}p} \\ \mathbf{0}_{m,1} \end{bmatrix}.$$

Here $\boldsymbol{R}_s^{\boldsymbol{x}p}$ is the $n \times 1$ vector of correlations of $Y^s(\boldsymbol{x}, \boldsymbol{\Theta})$ with $Y^s(\boldsymbol{x}_i^p, \boldsymbol{\Theta})$ which has i^{th} component $R_s\big((\boldsymbol{x}, \boldsymbol{\Theta}), (\boldsymbol{x}_i^p, \boldsymbol{\Theta})\big)$, $i = 1, \ldots n$; $\boldsymbol{R}_s^{\boldsymbol{x}s}$ is the $m \times 1$ vector of correlations of $Y^s(\boldsymbol{x}, \boldsymbol{\Theta})$ with $Y^s(\boldsymbol{x}_j^s, \boldsymbol{t}_j)$ which has j^{th} component $R_s\big((\boldsymbol{x}, \boldsymbol{\Theta}), (\boldsymbol{x}_j^s, \boldsymbol{t}_j)\big)$, $j = 1, \ldots m$; $\boldsymbol{R}_\delta^{\boldsymbol{x}p}$ is the $n \times 1$ vector of correlations of $\Delta(\boldsymbol{x})$ with $\Delta(\boldsymbol{x}_i^p)$ having i^{th} component $R_\delta(\boldsymbol{x}, \boldsymbol{x}_i^p)$, $i = 1, \ldots n$.

Level $100 \times (1-\alpha)\%$ uncertainty bounds for $\widehat{\mu}(\boldsymbol{x})$ are obtained by determining the lower and upper $\alpha/2$ quantiles of the draws $E[M(\boldsymbol{x}) \mid \mathcal{Y}, \boldsymbol{\Omega}^q]$, $q = 1, \ldots, N_{mcmc}$.

8.3.3 Bayesian Inference for the Bias $\delta(\boldsymbol{x})$ and Calibrated Simulator $E[Y^s(\boldsymbol{x}, \boldsymbol{\Theta}) \mid \mathcal{Y}]$

Often it is of interest to predict the bias function to measure the magnitude of the simulator code inaccuracies over the domain of the control inputs as well as to identify those control variables (and their values) that produce the largest biases. Bayesian inference about $\delta(\boldsymbol{x})$ is based on examining the posterior distribution $[\Delta(\boldsymbol{x}) \mid \mathcal{Y}]$. In particular, the mean of the posterior, i.e.,

$$\widehat{\delta}(\boldsymbol{x}) = E\left[\Delta(\boldsymbol{x}) \mid \mathcal{Y}\right]$$

can be taken as a Bayesian prediction of $\delta(\boldsymbol{x})$.

A strategy similar to that for computing $\widehat{\mu}(\boldsymbol{x})$ can be used to determine

$$\begin{aligned} \widehat{\delta}(\boldsymbol{x}) &= E\left[\Delta(\boldsymbol{x}) \mid \mathcal{Y}\right] \\ &= \int E\left[\Delta(\boldsymbol{x}) \mid \mathcal{Y}, \boldsymbol{\Omega}\right] \, [\boldsymbol{\Omega} \mid \mathcal{Y}] \; d\boldsymbol{\Omega} \end{aligned}$$

$$\approx \frac{1}{N_{mcmc}} \sum_{q=1}^{N_{mcmc}} E\left[\varDelta(\boldsymbol{x}) \mid \mathcal{Y}, \boldsymbol{\varOmega}^q\right] , \qquad (8.3.6)$$

where $\boldsymbol{\varOmega}^q = (\beta_0^q, \boldsymbol{\lambda}^q, \boldsymbol{\rho}^q, \boldsymbol{\varTheta}^q)$, $q = 1, \ldots, N_{mcmc}$, are independent draws from $[\boldsymbol{\varOmega} \mid \mathcal{Y}]$. The conditional mean in (8.3.6) can be obtained using an analogous argument to that for $E[M(\boldsymbol{x}) \mid \mathcal{Y}, \boldsymbol{\varOmega}]$ after observing that $[\varDelta(\boldsymbol{x}), \mathcal{Y} \mid \boldsymbol{\varOmega}]$ is multivariate normally distributed. The conditional mean of $\varDelta(\boldsymbol{x})$ given $\mathcal{Y}$ and $\boldsymbol{\varOmega}$ is

$$E\left[\varDelta(\boldsymbol{x}) \mid \mathcal{Y}, \boldsymbol{\varOmega}\right] = \lambda_\delta^{-1} \left(\boldsymbol{R}_\delta^{xp}\right)^\top \boldsymbol{\varSigma}_{\mathcal{Y}}^{-1} \left(\mathcal{Y} - \beta_0 \mathbf{1}_{n+m}\right) \qquad (8.3.7)$$

which is substituted into (8.3.6). The quantiles of the draws $E[\varDelta(\boldsymbol{x}) \mid \mathcal{Y}, \boldsymbol{\varOmega}^q]$ for $q = 1, \ldots, N_{mcmc}$ provide uncertainty bounds for $\widehat{\delta}(\boldsymbol{x})$.

The quantity

$$E\left[Y^s(\boldsymbol{x}, \boldsymbol{\varTheta}) \mid \mathcal{Y}\right] = E\left[E\left[Y^s(\boldsymbol{x}, \boldsymbol{\varTheta}) \mid \mathcal{Y}, \boldsymbol{\varOmega}\right]\right]$$

is called the *calibrated simulator* because it can be thought of as a predictor of the simulator at $\boldsymbol{x}$ when evaluated at the true value of the calibration parameter, i.e., of $y^s(\boldsymbol{x}, \boldsymbol{\theta})$. The unknown $\boldsymbol{\theta}$ is replaced by an average over the posterior $\boldsymbol{\varTheta}$ draws, and the unknown $y^s(\boldsymbol{x}, \boldsymbol{t})$ is predicted using posterior draws of the remaining parameters. Arguments similar to those in Eqs. (8.3.6) and (8.3.7) give an MCMC-based estimate of the calibrated simulator as well as an uncertainty quantification of this value.

The remainder of this chapter will first illustrate the Bayesian methodology presented above using the analytic Example 8.1 and then using the injection molding Example 1.4 in Sect. 1.2.

Example 8.1 *(Continued)*. In this example there is no value of the calibration parameter t that produces a simulator with zero error for all $x \in [-5, 5]$, i.e., there is no t for which

$$\max\{\, |\, \mu(x) - y^s(x, t)\, | : \; x \in [-5, +5]\,\} = 0\,.$$

Thus this example mimics applications where simplified physics or biology can result in an imperfect simulator.

While the primary purpose of this example is to illustrate the results of a Bayesian analysis in which "expert opinion" about θ is available, a short digression will first be made by considering a value of t based on a metric that measures the difference between $\mu(x)$ and $y^s(x, t)$, i.e., using engineering validation. Two simple metrics select t to minimize

$$\max\{\, |\, \mu(x) - y^s(x, t)\, | : \; x \in [-5, +5]\,\} \;\text{ or }\; \int_{-5.0}^{+5.0} (\mu(x) - y^s(x, t))^2 \, dx\,. \qquad (8.3.8)$$

Using the true $\mu(x)$ and $y^s(x, t)$, calculation shows that $t = 1.23$ minimizes the L_2 distance on the right-hand side of (8.3.8); i.e., $y^s(x, +1.23)$ is the L_2-calibrated predictor of $\mu(x)$. Figure 8.3 shows that $y^s(x, 1.23)$ is a reasonably accurate simulator of $\mu(x)$ for all $x \in [-5.0, 5.0]$ with the exception of x near $+5.0$ where the bias, $\mu(x) - y^s(x, 1.23)$, is greatest. Of course, in practice, neither $\mu(x)$ nor $y^s(x, t)$ is known.

One data-based L_2 calibration parameter is

$$\hat{t} = \arg\min_t \sum_{\ell=1}^{5} \left(y^p(x_\ell^p) - \widehat{y^s}(x_\ell^p, t) \right)^2$$

which leads to the predictor $\widehat{\mu}(x) = \widehat{y^s}(x, \hat{t}\,)$ of $\mu(x)$ where $\widehat{y^s}(x, t)$ is an EBLUP of $y^s(x, t)$. This method of estimating t replaces $\mu(x)$ by $y^p(x)$, the L_2 integral by the sum over the five x_ℓ^p used to conduct the physical system experiment, and $y^s(x, t)$ by $\widehat{y^s}(x, t)$.

Returning to the primary objective of this example, Bayesian calibration, the user must first identify a prior distribution for the model parameters, $\boldsymbol{\Omega} = (\lambda_\epsilon, \lambda_s, \lambda_\delta, \rho^x, \rho^t, \rho^\delta, \Theta\big)$. To simplify the analysis, this example will use independent priors for each parameter. The precision parameters $(\lambda_\epsilon, \lambda_s, \lambda_\delta)$ will be given gamma priors; the correlation parameters $(\rho^x, \rho^t, \rho^\delta)$ as well as the calibration parameter Θ will be given beta priors. Two data transformations are introduced to streamline the interpretability of certain prior parameters. The first transformation standardizes the output from the simulator runs and the physical experiments. This transformation will help specify the precision parameters. The simulator output is normalized by

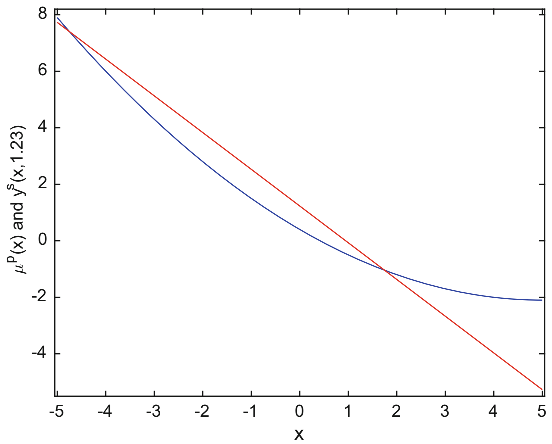


Fig. 8.3 The mean $\mu(x) = 0.1x^2 - x + 0.4$ (blue) and the L_2 predictor $y^s(x, 1.23) = 1.23 - 1.3x$ (red)

$$y^s_{std}(x_i^s, t_i) = \frac{y^s(x_i^s, t_i) - \bar{y}^s}{s(y^s)}, \quad i = 1, \ldots, m, \tag{8.3.9}$$

where

$$\bar{y}^s = \frac{1}{m} \sum_{i=1}^{m} y^s(x_i^s, t_i)$$

is the sample mean over the m simulation runs and $s(y^s)$ is the sample standard deviation of these values. The transformed $y^s_{std}(\cdot,\cdot)$ have *sample mean* of zero and *sample variance* of unity. The *physical* system output is standardized using the *simulator* values

$$y^p_{std}(x^s_i) = \frac{y^p(x^s_i) - \bar{y}^s}{s(y^s)}, \quad i = 1, \ldots, n\,. \tag{8.3.10}$$

Thus the conditional $Y^s(x,t)$ process that is fit to the standardized data is assumed to have *process mean that is zero and process variance, $1/\lambda_s$, that is (near) unity*. Table 8.3 shows that a gamma prior distribution is assumed for the error precision, λ_ϵ, which has mean 100. The value 100 corresponds to an associated measurement error standard deviation, σ_ϵ, of 0.1. A $\sigma_\epsilon = 0.1$ is 10% of the assumed unit process standard deviation σ_s of $Y^s(x,t)$.

Second, a transformation is performed on the *inputs* to facilitate specification of the priors for ρ^x, ρ^t, and ρ^δ. By subtracting the minimum value of $t_1, \ldots, t_m$ from each t_i and dividing each difference by the range of $t_1, \ldots, t_m$, the calibration inputs for the simulator are modified to span $[0,1]$. Analogous transformations are performed for $x^s_1, \ldots, x^s_m$ and $x^p_1, \ldots, x^p_n$ jointly to put each of the three sets of inputs on $[0,1]$. By examining Eq. (8.2.1), for example, this transformation implies that ρ^x is the correlation between $Y^s(x_1,t_1)$ and $Y^s(x_2,t_2)$ when $t_1 = t_2$ and $|x_1 - x_2| = 1$, i.e., are the extremes of the control input range. Expert opinion about the association between such pairs of output can then be assessed. The correlation parameters are assumed independent with common prior $Be(5,5)$; this prior has mean 0.5 and substantial support (prior probability 0.96) over $(0.2, 0.8)$. These assumptions are summarized in Table 8.3.

Parameter	Prior	Parameter	Prior
λ_ϵ	$\Gamma(1, 0.01)$	λ_s	$\Gamma(5,5)$
ρ^x	$Be(5,5)$	ρ^t	$Be(5,5)$
ρ^δ	$Be(5,5)$	Θ	$TrN(0.50, 0.25)$

Table 8.3 Prior distributions used in Example 8.1. The notation $\Gamma(a,b)$ is defined by (B.2.1); the notation $Be(a,b)$ is defined by (B.3.1); $TrN(0.50,(0.25)^2)$ denotes the univariate normal distribution with mean 0.50 and standard deviation 0.25 that is truncated to (0,1)

Let (a_δ, b_δ) denote the parameters for the gamma prior of λ_δ. This example uses *three* (a_δ, b_δ) to study the impact of the λ_δ prior; they are listed in Table 8.4. The intuition about the impact of these priors is as follows. Given λ_δ, the standard deviation of the $\Delta(x)$ process is $1/\sqrt{\lambda_\delta}$; draws from a $\Delta(x)$ having larger process standard deviation will allow larger bias values. The effect of prior on the $\Delta(x)$ draws can be intuited by observing that the mean value of the λ_δ values is $E[\lambda_\delta] = a_\delta/b_\delta$. Thus the $\delta(x)$ will (stochastically) have *greater range* for *larger b_δ/a_δ*. Table 8.4 has arranged the three priors by increasing $1/\sqrt{E[\lambda_\delta]}$ value.

The Metropolis–Hastings MCMC with adaptive training of proposal widths using Roberts and Rosenthal (2007) was used in the Bayesian inference below. The

Prior	a_δ	b_δ	$E[\lambda_\delta] = a_\delta/b_\delta$	$Var(\lambda_\delta) = a_\delta/(b_\delta)^2$	$\sigma_\delta \approx 1/\sqrt{E[\lambda_\delta]}$
1	1300	2.08	625	300	0.04
2	12	2	6	3	0.41
3	0.5	2	0.25	0.125	2.00

Table 8.4 The three $\Gamma(a_\delta, b_\delta)$ prior distributions for λ_δ used in Example 8.1

first 5000 $\boldsymbol{\Omega}$ draws were discarded as burn-in followed by 10,000 production draws. Every 20th of the production runs was kept for inference, providing a total of 500 draws from the joint posterior of $\boldsymbol{\Omega}$.

The effect of the three λ_δ priors on the Bayesian inference can now be examined. The left panels of Figs. 8.4, 8.5, and 8.6 plot the posterior draws of $1/\sqrt{\lambda_\delta}$. These histograms use different horizontal scales to better show the effect of the prior. In all cases the mean of the $1/\sqrt{\lambda_\delta}$ posterior draws is very near $1/\sqrt{E[\lambda_\delta]}$, with these values being approximately 0.04, .40, and 2.0 for Priors 1, 2, and 3, respectively.

The smaller the posterior $1/\sqrt{\lambda_\delta}$ draws, the smaller the estimated $\delta(x)$ and its uncertainty limits (at any fixed x). The left panels of Figs. 8.7, 8.8, and 8.9 show the differences in these quantities for Priors 1, 2, and 3. For Prior 1, the estimated $\delta(x)$ is nearly zero with small 90% uncertainty bounds, while the estimated $\delta(x)$ has range -2 to $+4$ for Prior 3 with 90% credible limits on the order of three units above and below the estimated $\delta(x)$. The right panels of Figs. 8.7, 8.8, and 8.9 show the cumulative effect of the prior on the predicted $\mu(x)$. For Prior 1, with nearly zero estimated $\delta(x)$, the predicted $\mu(x)$ is essentially the averaged straight line,

$$\widehat{\mu}(x) \approx E\left[Y^s(x, \Theta) \mid \mathcal{Y}\right].$$

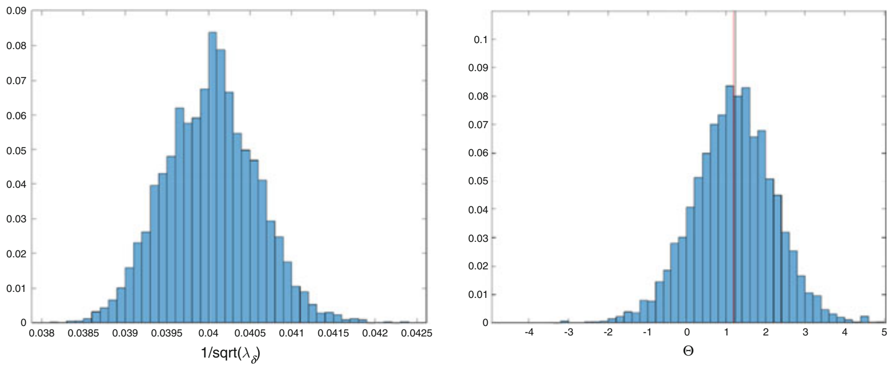


Fig. 8.4 **For Prior 1** *Left panel*: histogram of 500 $1/\sqrt{\lambda_\delta}$ values based on posterior draws from $[\lambda_\delta \mid \mathcal{Y}]$. *Right panel*: histogram of 500 draws from the posterior distribution $[\Theta \mid \mathcal{Y}]$ of the calibration parameter; the vertical black line $t = 1.23$ is the L_2 calibration parameter and the vertical red line $t = 1.19$ is the mean of the $[\Theta \mid \mathcal{Y}]$ draws

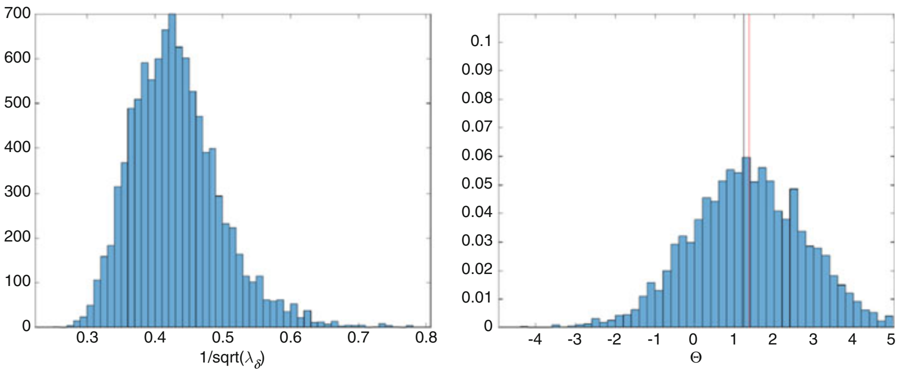


Fig. 8.5 **For Prior 2** *Left panel*: histogram of 500 $1/\sqrt{\lambda_\delta}$ values based on posterior draws from $[\lambda_\delta \mid \mathcal{Y}]$. *Right panel*: histogram of 500 draws from the posterior distribution $[\Theta \mid \mathcal{Y}]$ of the calibration parameter; the vertical black line $t = 1.23$ is the L_2 calibration parameter, and the vertical red line $t = 1.37$ is the mean of the $[\Theta \mid \mathcal{Y}]$ draws

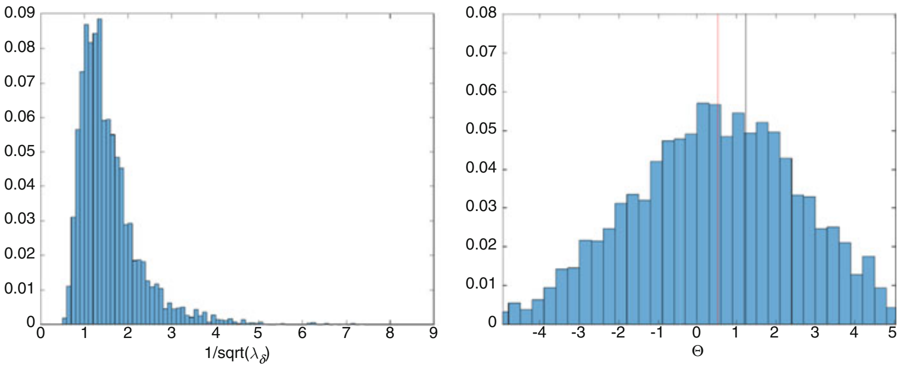


Fig. 8.6 **For Prior 3** *Left panel*: histogram of 500 $1/\sqrt{\lambda_\delta}$ values based on posterior draws from $[\lambda_\delta \mid \mathcal{Y}]$. *Right panel*: histogram of 500 draws from the posterior distribution $[\Theta \mid \mathcal{Y}]$ of the calibration parameter; the vertical black line $t = 1.23$ is the L_2 calibration parameter and the vertical red line $t = 0.52$ is the mean of the $[\Theta \mid \mathcal{Y}]$ draws

In contrast the "large" posterior $1/\sqrt{\lambda_\delta}$ draws in Prior 3 result both in a large estimated $\delta(x)$ and a predicted $\mu(x)$ which slavishly follows the physical observations. For Prior 2 the posterior $1/\sqrt{\lambda_\delta}$ draws produce a predicted $\mu(x)$ which is nearly equal to the target mean of the physical system.

The impact of the prior on the predicted θ is less pronounced than on $\delta(x)$. Histograms of the 500 inference draws from the posterior $[\Theta \mid \mathcal{Y}]$ are shown in the right panels of Figs. 8.4, 8.5, and 8.6. Table 8.5 lists the median and 90% uncertainty bounds based on these 500 draws. The width of the uncertainty bounds for θ

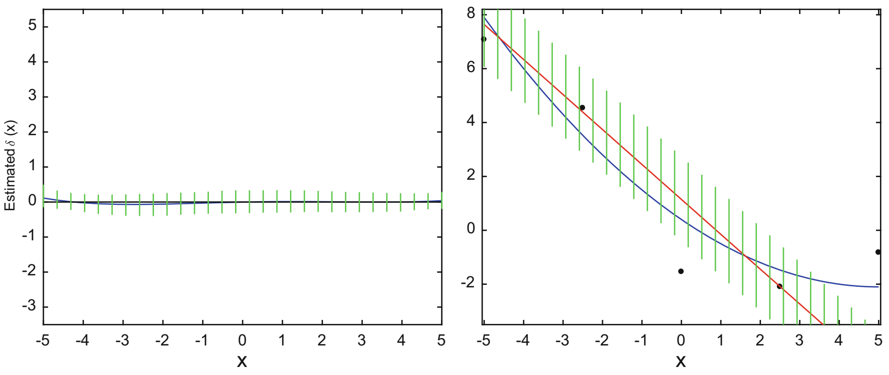


Fig. 8.7 For Prior 1 *Left panel*: expected bias $E[\varDelta(x) \mid \mathcal{Y}]$ of the simulator at θ and 90% uncertainty bounds. *Right panel*: the target function $\mu(x) = 0.1x^2 - x + 0.4$ (blue); the $n = 5$ physical system observations $y^p(x)$ from Table 8.2 (black); the Bayesian-calibrated predictor $\widehat{\mu}(x)$ in (8.3.5) (red); 90% uncertainty bounds for the estimated $\mu(x)$ at 30 equally spaced x (green)

increases with the prior range of the $\varDelta(x)$ draws, i.e., it increases for Priors 1, 2, and 3. The bounds are narrowest for Prior 1 and widest for Prior 3. The central posterior $\varTheta$ values, a Bayesian prediction of θ, are somewhat stable and, for Priors 1 and 2, favor a θ prediction which is a bit over 1.0; this conclusion is consistent with the minimum L_2 choice of calibration parameter, $t = 1.23$.

Prior	Estimated θ	Uncertainty bounds
1	1.26	(−0.52, 2.81)
2	1.40	(−1.06, 3.69)
3	0.52	(−3.00, 3.86)

Table 8.5 Bayesian prediction of θ with 90% uncertainty bounds

In sum, this example shows the importance of the careful specification of the prior for calibration problems. Calibration based on unjustified priors can lead to substantially inaccurate predictions and uncertainty quantification. ◆

Example 1.4 (Continued). Recall that this example studies an injection molding (IM) process in which a melted thermoplastic polyolefin is injected into a form. The goal of the example is to show how calibration can be performed for a real-valued outcome; specifically, the calibration is performed to combine simulator and physical experiment output to improve prediction of the relative shrinkage of the *mean width of a test part* as a function of four control variables.

The data to perform this calibration were obtained from experiments involving a physical IM system measured at 19 control variable settings (see Table 1.2). Additional data were obtained from a computational simulator of the IM process which

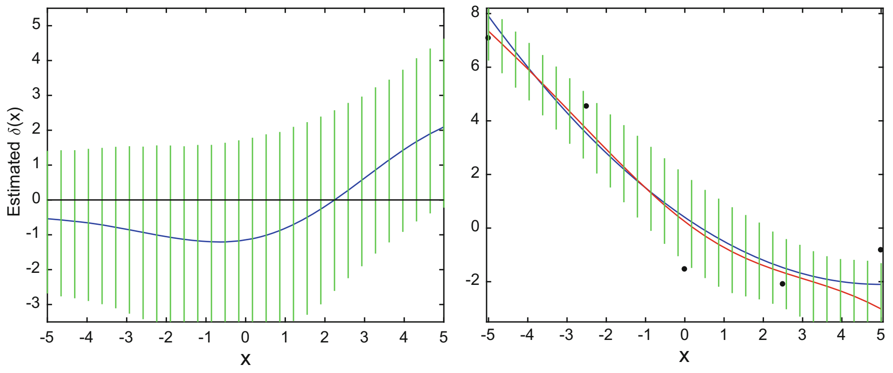


Fig. 8.8 **For Prior 2** *Left panel*: expected bias $E[\Delta(x) \mid \mathcal{Y}]$ of the simulator at θ and 90% uncertainty bounds. *Right panel*: the target function $\mu(x) = 0.1x^2 - x + 0.4$ (blue); the $n = 5$ physical system $y^p(x)$ from Table 8.2 (black); the Bayesian-calibrated predictor $\widehat{\mu}(x)$ in (8.3.5) (red); 90% uncertainty bounds for the estimated $\mu(x)$ at 30 equally spaced x (green)

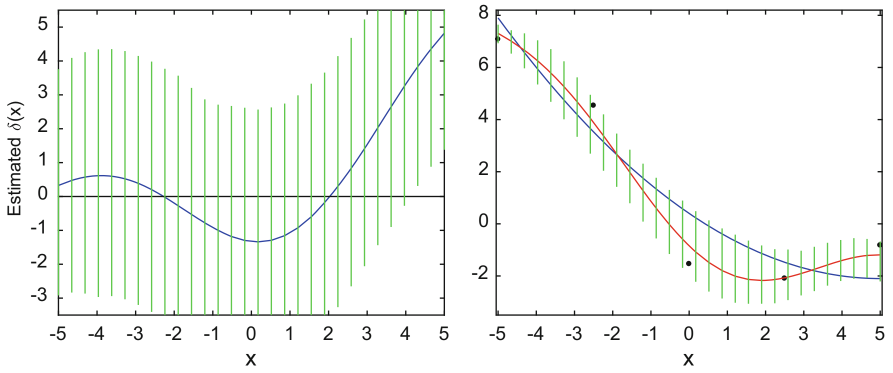


Fig. 8.9 **For Prior 3** *Left panel*: expected bias $E[\Delta(x) \mid \mathcal{Y}]$ of the simulator at θ and 90% uncertainty bounds. *Right panel*: the target function $\mu(x) = 0.1x^2 - x + 0.4$ (blue); the $n = 5$ physical system observations $y^p(x)$ from Table 8.2 (black); the Bayesian-calibrated predictor $\widehat{\mu}(x)$ in (8.3.5) (red); 90% uncertainty bounds for the estimated $\mu(x)$ at 30 equally-spaced x (green)

calculated the dimensions of a test part produced by arbitrary user-specified values of the control variables. In addition to the four control inputs, the simulator required specification of three heat transfer coefficients for the mold during the flow, packing, and cooling phases of fabrication. The heat transfer coefficients are difficult to measure during the IM process and were used as calibration parameters for this example. Measured relative width shrinkages were obtained from $m = 35$ simulator runs based on a 35×7 space-filling design (see Fig. 1.6).

As in the previous example, Bayesian calibration will be performed using the residuals from normalized versions of the simulator and physical experiment data. Using Eqs. (8.3.9) and (8.3.10), the simulator and the physical experiment data are both standardized by the mean and standard deviation of the *simulator data*. The left and right panels of Fig. 8.10 plot the relative shrinkage and normalized relative shrinkage, respectively, of the width of the test pieces versus t_{Pack}, which will be seen below to be the most active input.

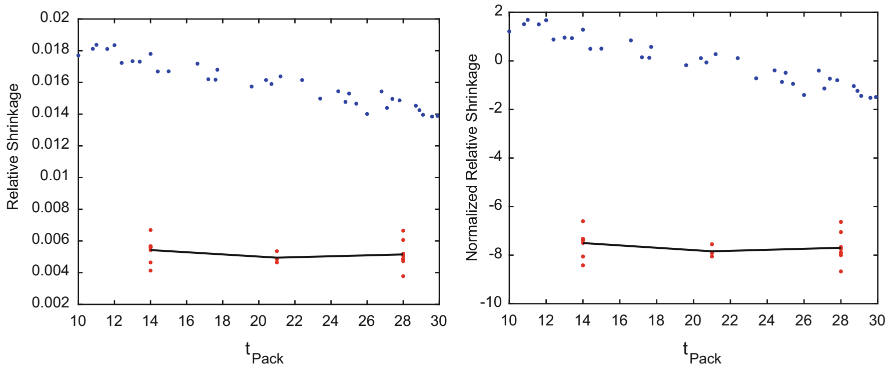


Fig. 8.10 Left panel: scatterplot of the relative shrinkages for the 35 simulator runs versus the packing time, t_{Pack}, of the IM manufacturing process (blue); plot of observed relative shrinkages for the 19 physical experiment outputs versus t_{Pack} (red). Right panel: scatterplot of normalized relative shrinkages for the 35 simulator runs and the 19 physical experiment outputs

In addition to plotting the data, a sensitivity analysis is performed to gain additional understanding of these data. The sensitivity analysis was used to determine the relative activity of the seven inputs on the simulated values of the normalized relative shrinkage (see Chap. 7). The main effect and total effect sensitivity indices (SIs) from this computation are listed in Table 8.6. As measured by the total effect SI, the most important control inputs are the packing time, t_{Pack}, and the packing pressure, P_{Pack}, with t_{Pack} being the most important by far. Compared with the control inputs, none of the three HTCs is important; compared among themselves, HTC$_{open}$ is the most important HTC. In future prediction of the mean, T_{melt} and t_{Cool} will be fixed, and slices of predictions of the mean surface will be plotted for varying t_{Pack} and P_{Pack}.

As opposed to using subject-matter experts to suggest priors for the parameters of the calibration model, this example will use independent and heuristically selected

Variable	ME SI	TE SI
T_{melt}	0.0024	0.0044
t_{Pack}	0.9191	0.9234
P_{Pack}	0.0627	0.0639
t_{Cool}	0.0101	0.0118
HTC$_{flow}$	0.0003	0.0004
HTC$_{pack}$	0.0000	0.0000
HTC$_{open}$	0.0006	0.0011

Table 8.6 Main effect (ME) and total effect (TE) sensitivity indices (SIs) for the control and calibration inputs to the simulator output for the relative shrinkage of the width of the Example 1.4 test piece (based on 35 runs)

parametric prior families for the model parameters. Where possible, the data will be used to suggest values for the hyperparameters.

Consider the prior for the bias precision λ_δ as illustrated in the right panel of Fig. 8.10. The scatterplot shows that, as t_{Pack} increases, the normalized relative shrinkage decreases slightly for test parts produced in the physical experiments; however, the normalized relative shrinkage is nearly constant for the simulator runs. Thus the bias appears to decrease modestly from a value of about 8.0 when $t_{Pack} = 14$ to about 5.0 when $t_{Pack} = 28$. Setting twice the conditional standard deviation of $\delta(\boldsymbol{x})$, $2/\sqrt{E[\lambda_\delta]}$, equal to 7.0 or equivalently $E[\lambda_\delta] = (2/7)^2 = 0.08$ allows bias magnitudes of sizes 5.0–8.0 to be easily attainable. As a second requirement used to specify the pair of hyperparameters, the prior probability requirement $P[0.018 \leq \lambda_\delta \leq 0.16] = 0.90$ is made. Thus the Bayesian analysis sets $(a_\delta, b_\delta) = (2.88,\ 36)$ which satisfies $2/\sqrt{E[\lambda_\delta]} = 7.07$ as well as the coverage requirement.

To suggest hyperparameters for the precision, λ_ϵ, of the measurement error in the normalized relative shrinkages, a regression analysis of the normalized shrinkages was conducted. The fitted mean of the regression was the main effects model in the four control variables; 19 observations from the manufacturing system were used to fit the regression. The estimated residual mean squared error was 0.80. The Bayesian prior for λ_ϵ was taken to be the $\Gamma(0.08, 0.02)$ distribution. Thus the mean λ_ϵ was assumed to be 4, and (conditional) draws from the associated $\epsilon(\boldsymbol{x})$ process can be as large as $2/\sqrt{4} = 1.0$ but can adapt substantially because this pair of hyperparameters was also selected to satisfy $P[0 \leq \lambda_\epsilon \leq 100] = 0.90$.

All correlation parameters were taken to have $Be(\alpha,\beta)$ prior distributions with hyperparameters (α,β). The hyperparameters for all ρ^x were taken to be the somewhat neutral value of $(\alpha,\beta) = (5, 5)$. The ρ^δ hyperparameters were set equal to $(5, 1)$ reflecting the belief that the simulator bias is relatively constant. Lastly, the ρ^t hyperparameters were also set equal to $(5, 5)$. In sum, Table 8.7 lists the prior parameters for the marginal priors adopted for this example.

A Metropolis–Hastings MCMC was run with 5000 burn-in $\boldsymbol{\Omega}$ runs and 10,000 production runs. Every 20th production run was used for the inferences below, for a total of 500 posterior $[\boldsymbol{\Omega} \mid \mathcal{Y}]$ values.

The inferences for the Bayesian analysis based on the 500 $[\boldsymbol{\Omega} \mid \mathcal{Y}]$ draws start with Fig. 8.11 which is a histogram of the 500 draws of conditional standard deviation of $\delta(\boldsymbol{x})$, $1/\sqrt{\lambda_\delta}$, obtained from the $\boldsymbol{\Omega}$ posterior. These values roughly range from 2.0 to 8.0 and are consistent with the prior intuition given by examining Fig. 8.10.

Parameter	Prior	Parameter	Prior
λ_ϵ	$\Gamma(0.08, 0.02)$	λ_s	$\Gamma(5, 5)$
$\rho^x_1, \rho^x_2, \rho^x_3, \rho^x_4$	i.i.d $Be(5, 5)$	$\rho^t_1, \rho^t_2, \rho^t_3$	i.i.d $Be(5, 5)$
λ_δ	$\Gamma(2.88, 36)$	$\rho^\delta_1, \rho^\delta_2, \rho^\delta_3, \rho^\delta_4$	i.i.d $Be(5, 1)$
Θ_1, Θ_2, Θ_3	i.i.d $TrN(0.50, 10)$		

Table 8.7 Independent marginals of the prior distribution used in Example 1.4. The notation is the same as used in Table 8.3

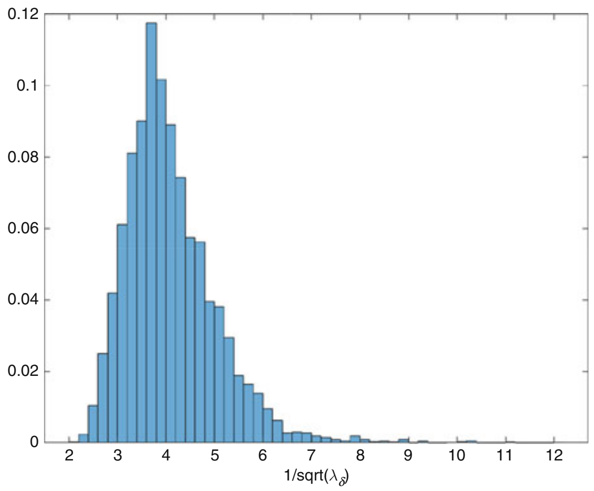


Fig. 8.11 Histogram of $1/\sqrt{\lambda_\delta}$ values from 500 posterior draws of λ_δ

Turning attention to inferences concerning the calibration parameters, Fig. 8.12 is a gray-scale histogram of the marginal posterior draws of the HTCs, a pairwise joint scatterplot of these draws, and 50 and 90% empirical joint bounds (in blue) for each pair of HTCs. The uniformly distributed marginal plots show that all three coefficients are not well specified, a finding that is consistent with the sensitivity analysis which suggested none of the HTCs had great influence on the simulated relative shrinkage. The joint posterior plot shows the pairs appear uncorrelated.

The estimated bias function for the simulator output, on the unnormalized scale, is shown in Fig. 8.13 over a grid of (t_{Pack}, P_{Pack}) values for fixed $(T_{melt}, t_{Cool}) = (200, 43)$. The bias is nearly linear. It is constant in P_{Pack} and increases as the packing time, t_{Pack}, increases to a maximum of 6.5×10^{-3}.

Lastly, calibrated predictions of the mean of the physical output, $\mu(T_{melt}, t_{Pack}, P_{Pack}, t_{Cool})$, were made for the 19 physical system inputs $(T_{melt}, t_{Pack}, P_{Pack}, t_{Cool})$. Figure 8.14 is a scatterplot of these 19 predictions of $\mu(T_{melt}, t_{Pack}, P_{Pack}, t_{Cool})$ versus the observed $y(T_{melt}, t_{Pack}, P_{Pack}, t_{Cool})$. The scatter above and below the predicted mean shows the measurement errors, and their distribution implies the effectiveness of the bias correction across the spectrum of the observed inputs. ♦

We conclude this section on the importance of the prior distribution in Bayesian calibration by noting that Brynjarsdóttir and O'Hagan (2014) provide an additional example of showing the importance of accounting for model discrepancy by using a realistic prior in order to produce posterior distributions that cover unknown calibration parameters.

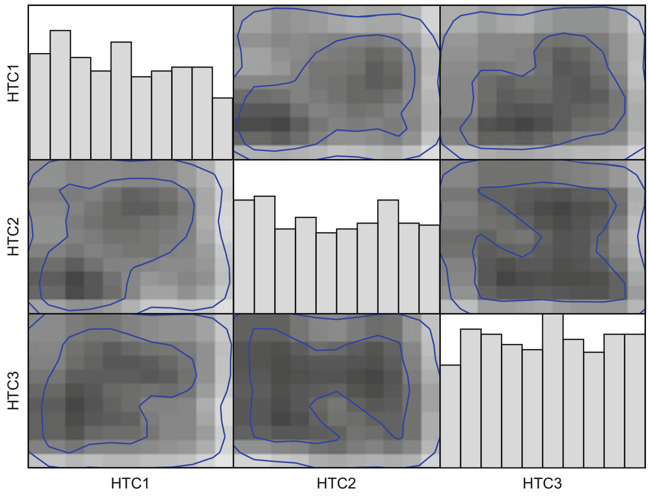


Fig. 8.12 Gray-scale scatterplot of the joint posterior distribution of the mold heat transfer coefficients during the flow, packing, and cooling phases of manufacturing; empirical 50 and 90% uncertainty sets for the joint values

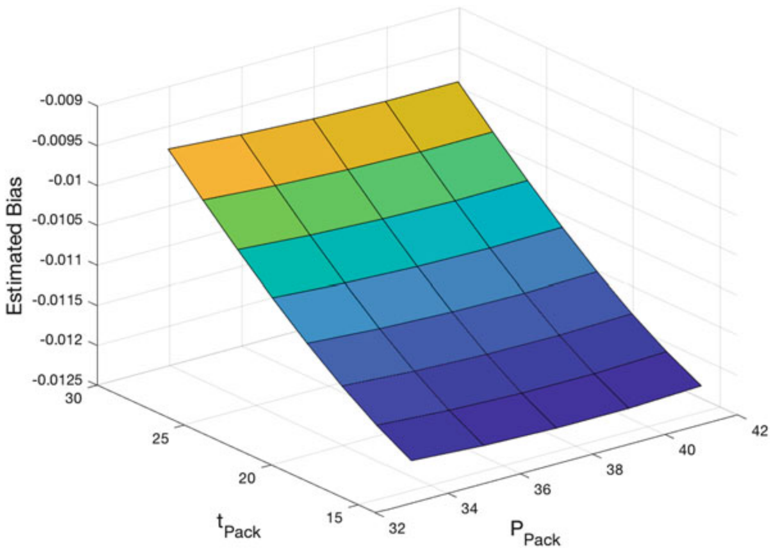


Fig. 8.13 Estimated bias function $\delta(T_{melt}, t_{Pack}, P_{Pack}, t_{Cool})$ of the calibrated predictor of the mean relative shrinkage of the width of a test piece for fixed $(T_{melt}, t_{Cool}) = (200, 43)$

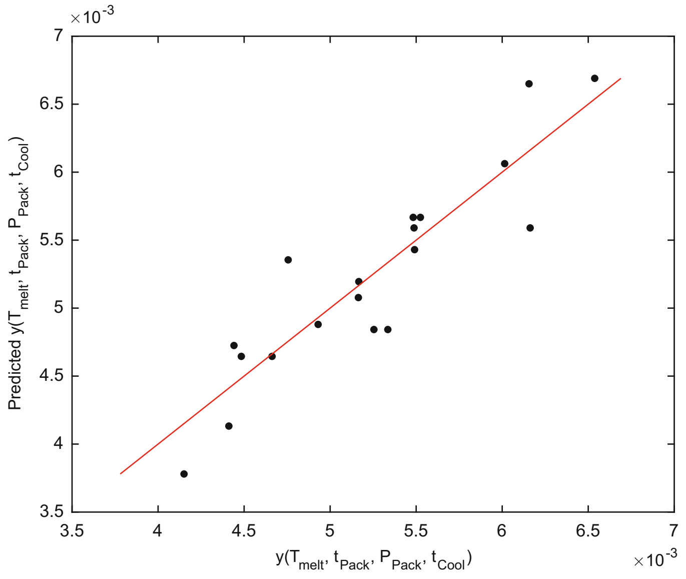


Fig. 8.14 Predicted mean relative shrinkage of the width of a test piece, $\mu(T_{melt}, t_{Pack}, P_{Pack}, t_{Cool})$, for the 19 observed inputs versus the observed values of $y(T_{melt}, t_{Pack}, P_{Pack}, t_{Cool})$. The reference line $y = x$ is shown in red

8.4 Calibration with Functional Data

This section describes an extension of the KOH model to settings in which the simulation code (simulator) produces either functional or multivariate output data. The proposed model is rather robust in that it includes numerical nugget effects to enhance fitting of an emulator to the simulator and simulator bias errors.

Transformations to the simulator and experimental data that enhance the Bayesian prior specification are given in Sects. 8.4.1 and 8.4.2. Section 8.4.3 provides expressions for the log likelihood of both types of data; the final formulas for the log likelihood are given by (8.4.51) and (8.4.67) for the cases of calibration in which the simulator is allowed to differ from the true mean model of the physical data and in which any such differences are assumed negligible, respectively. Section 8.5 provides a prior and methods to generate samples from the resulting posterior. It describes methodology to use the posterior samples at unsampled inputs for the following tasks:

Case 1: emulate the simulation output using only simulator data,
Case 2: emulate the calibrated simulator output modeling the simulator bias, and
Case 3: emulate the calibrated simulator output assuming no simulator bias.

More detail is given in this section and Sect. 8.5 than in other sections of the book to provide the interested reader with the information required to apply the Bayesian calibration methodology described here.

8.4.1 The Simulation Data

Let $\boldsymbol{y}^s(\boldsymbol{x}^s, \boldsymbol{t})$ denote an $m_s \times 1$ vector of simulation output that is calculated at input setting $(\boldsymbol{x}^s, \boldsymbol{t})$; the length of the simulation output vector is assumed to be the same for all inputs. Each element of $\boldsymbol{y}^s(\boldsymbol{x}^s, \boldsymbol{t})$ is associated with one or more index variable(s) which is not included in $\boldsymbol{x}^s$. If the suppressed index is continuous, such as time, space, or space and time, the output is referred to as *functional*. If it is categorical, such as a response indicator, the output is referred to as *multivariate*. For example, the time traces in Example 1.7 or 1.8 would be considered functional data, while the collection of individual scalar outputs in Example 1.6 would be considered multivariate data. The following discussion generally follows the development of Higdon et al. (2008).

Suppose the simulator is run m times producing $\boldsymbol{y}^s(\boldsymbol{x}^s_1, \boldsymbol{t}_1), \ldots, \boldsymbol{y}^s(\boldsymbol{x}^s_m, \boldsymbol{t}_m)$. As in the univariate case, the statistical modeling of these data is facilitated by *standardizing* them through centering and scaling. Let $\boldsymbol{Y}^s$ denote the $m_s \times m$ matrix of simulation output arranged with one $m_s \times 1$ *column* for each input:

$$\boldsymbol{Y}^s = \left[\boldsymbol{y}^s(\boldsymbol{x}^s_1, \boldsymbol{t}_1) \cdots \boldsymbol{y}^s(\boldsymbol{x}^s_m, \boldsymbol{t}_m)\right] .$$

The output is centered by subtracting from each element of $\boldsymbol{Y}^s$ the mean of all the elements of $\boldsymbol{Y}^s$ in the same row, i.e.,

$$\overline{\boldsymbol{y}}^s = (1/m)\boldsymbol{Y}^s \boldsymbol{1}_m , \qquad (8.4.1)$$

where $\boldsymbol{1}_m$ is the $m \times 1$ vector of ones. This operation yields the centered data matrix,

$$\boldsymbol{Y}^{s,c} = \boldsymbol{Y}^s - \overline{\boldsymbol{y}}^s \boldsymbol{1}_m^\top = \boldsymbol{Y}^s \left(\boldsymbol{I}_m - (1/m)\, \boldsymbol{J}_m\right) ,$$

where $\boldsymbol{I}_m$ is the $m \times m$ identity matrix and $\boldsymbol{J}_m = \boldsymbol{1}_m \boldsymbol{1}_m^\top$ is the $m \times m$ matrix of ones. The sample covariance matrix of $\boldsymbol{Y}^s$, $\boldsymbol{C}^s$, is given by:

$$\boldsymbol{C}^s = \frac{1}{m-1} \boldsymbol{Y}^s \left(\boldsymbol{I}_m - (1/m)\boldsymbol{J}_m\right) (\boldsymbol{Y}^s)^\top .$$

Because all components of *functional data* have the *same* measurement unit, this data is typically scaled by the sample standard deviation ς^s of *all* the elements of $\boldsymbol{Y}^{s,c}$. The sample mean of the elements of $\boldsymbol{Y}^{s,c}$ is zero since, by construction, the m_s row means of $\boldsymbol{Y}^{s,c}$ are all zero. Hence the sample variance $(\varsigma^s)^2$ is proportional to the sum of squares of the $m_s \cdot m$ elements of $\boldsymbol{Y}^{s,c}$, which is given by $(m-1)\,\mathrm{tr}(\boldsymbol{C}^s)$, i.e., by

$$(\varsigma^s)^2 = \frac{m-1}{m_s \cdot m - 1} \operatorname{tr}(\boldsymbol{C}^s) \, . \tag{8.4.2}$$

Denote the $m_s \times m$ matrix of standardized functional output by

$$\boldsymbol{Y}^{s,n} = \frac{1}{\varsigma^s} \boldsymbol{Y}^{s,c} \, .$$

In contrast, the output components of *multivariate data* typically have *different* measurement units. In this case each of the m_s components of $\boldsymbol{Y}^{s,c}$ is scaled separately to obtain the standardized output

$$\boldsymbol{Y}^{s,n} = \left(\boldsymbol{C}^s_d\right)^{-1/2} \boldsymbol{Y}^{s,c} \, ,$$

where

$$\boldsymbol{C}^s_d = diag(c^s_{11}, c^s_{22}, \ldots, c^s_{m_s m_s}) \tag{8.4.3}$$

and c^s_{ii} is the i^{th} diagonal element of $\boldsymbol{C}^s$. The $diag(\cdot)$ operator produces a matrix with its scalar (or matrix) arguments placed in the diagonal entries (or blocks) and with zeros placed in every off-diagonal entry (or block).

A principal component decomposition is used frequently to derive a compact basis representation of the centered and scaled $\boldsymbol{y}^s(\boldsymbol{x}^s, \boldsymbol{t})$ output for the m-selected $(\boldsymbol{x}^s, \boldsymbol{t})$ inputs (other bases can be used). The singular value decomposition (SVD) of the standardized output matrix $\boldsymbol{Y}^{s,n}$ is a representation of the form

$$\boldsymbol{Y}^{s,n} = \boldsymbol{U}^s \boldsymbol{\Sigma}^s \left(\boldsymbol{V}^s\right)^\top \, ,$$

where $\boldsymbol{U}^s$ and $\boldsymbol{V}^s$ are $m_s \times m_s$ and $m \times m$ orthonormal matrices containing the left and right singular vectors of $\boldsymbol{Y}^{s,n}$, respectively, while $\boldsymbol{\Sigma}^s = \left(\Sigma^s_{ij}\right)$ is the $m_s \times m$ rectangular matrix whose diagonals are the corresponding singular values ordered from largest to smallest and whose off-diagonal values are zero. Given $p_s \le \min(m_s, m-1)$, the best rank p_s approximation to $\boldsymbol{Y}^{s,n}$ based on minimizing the Frobenius norm (see Stewart (1993)) is

$$\boldsymbol{U}^s \widetilde{\boldsymbol{\Sigma}}^s \left(\boldsymbol{V}^s\right)^\top \, ,$$

where $\widetilde{\boldsymbol{\Sigma}}^s$ is obtained from $\boldsymbol{\Sigma}^s$ by setting to zero all singular values less than the p_s largest. Because the squares of the singular values are the eigenvalues of the sample covariance matrix $\boldsymbol{C}^s$, the value of p_s is chosen to be the smallest integer not exceeding $\min(m_s, m-1)$ such that

$$\frac{\sum_{i=1}^{p_s} \left(\Sigma^s_{ii}\right)^2}{\sum_{i=1}^{\min(m_s, m-1)} \left(\Sigma^s_{ii}\right)^2} = \frac{\sum_{i=1}^{p_s} \left(\Sigma^s_{ii}\right)^2}{\operatorname{tr}(\boldsymbol{C}^s)} \ge \tau,$$

where τ is the desired fraction of the total variance, $\operatorname{tr}(\boldsymbol{C}^s)$, that is to be explained. Common choices of τ are 0.95 and 0.99.

Having determined the rank p_s of the approximation to the standardized data that explains a selected fraction of total variance, the matrices of the SVD are then partitioned as follows:

$$\boldsymbol{U}^s = \left[\boldsymbol{U}_1^s \; \boldsymbol{U}_2^s\right],$$

$$\boldsymbol{V}^s = \left[\boldsymbol{V}_1^s \; \boldsymbol{V}_2^s\right], \text{ and}$$

$$\widetilde{\boldsymbol{\Sigma}}^s = \begin{bmatrix} \widetilde{\boldsymbol{\Sigma}}_{11}^s & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix},$$

where $\boldsymbol{U}_1^s$ contains the first p_s columns of $\boldsymbol{U}^s$ and $\boldsymbol{U}_2^s$ contains the remaining columns, while $\boldsymbol{V}_1^s$ contains the first p_s columns of $\boldsymbol{V}^s$ and $\boldsymbol{V}_2^s$ contains the remaining columns, and lastly $\widetilde{\boldsymbol{\Sigma}}_{11}^s$ is the $p_s \times p_s$ upper-left block diagonal matrix of $\widetilde{\boldsymbol{\Sigma}}^s$. Define the matrix

$$\boldsymbol{K}^s = (1/\sqrt{m})\, \boldsymbol{U}_1^s \widetilde{\boldsymbol{\Sigma}}_{11}^s. \tag{8.4.4}$$

Then the best rank p_s approximation to the standardized data matrix $\boldsymbol{Y}^{s,n}$ is

$$\boldsymbol{K}^s \left(\sqrt{m}\, \boldsymbol{V}_1^s\right)^\top.$$

In other words, the i-th standardized output vector $\boldsymbol{y}^{s,n}(\boldsymbol{x}_i^s, \boldsymbol{t}_i)$ can be approximated by a linear combination of the orthogonal column vectors of $\boldsymbol{K}^s$, i.e., by

$$\boldsymbol{y}^{s,n}(\boldsymbol{x}_i^s, \boldsymbol{t}_i) \approx \sum_{j=1}^{p_s} \boldsymbol{k}_j^s \, w_j^s(\boldsymbol{x}_i^s, \boldsymbol{t}_i), \quad i = 1, \ldots, m,$$

where $\left(w_1^s(\boldsymbol{x}_i^s, \boldsymbol{t}_i), \ldots, w_{p_s}^s(\boldsymbol{x}_i^s, \boldsymbol{t}_i)\right)$ is the i^{th} row of $\left(\sqrt{m}\, \boldsymbol{V}_1^s\right)$, i.e., the i^{th} column of $\left(\sqrt{m}\, \boldsymbol{V}_1^s\right)^\top$, and $\boldsymbol{k}_j^s$ is the j-th column of $\boldsymbol{K}^s$. Because the columns of $\boldsymbol{V}_1^s$ are orthonormal and the coordinates of each column must sum to zero due to standardization of the output, the sample means and variances of $\{w_j^s(\boldsymbol{x}_i^s, \boldsymbol{t}_i)\}_{i=1}^m$ are 0 and 1, respectively, for all $j = 1, \ldots, p_s$.

Example 1.7 *(Continued)*. Recall that flyer plate experiments measure the velocity profile of a shock wave that is forced through a stationary target sample of a material, in this example tantalum. Initially, this example studies the output from $m = 128$ runs of a simulator of the velocity profile that varied $d = 10$ material properties and experimental conditions (Table 1.4) according to an orthogonal array-based Latin hypercube design (Sect. 5.3). The right panel of Fig. 1.12 plots the 128 velocity profiles.

Figure 8.15 shows the $p_s = 3$ unscaled eigenvectors (columns of $\varsigma^s \boldsymbol{K}^s$) corresponding to the three principal components selected by setting $\tau = 0.95$, computed from the full set of the 128 standardized simulation runs constituting the columns of the matrix $\boldsymbol{Y}^{s,n}$. ♦

The representation of the standardized output in terms of the orthogonal columns of a fixed matrix $\boldsymbol{K}^s$ weighted by coefficients that have sample mean 0 and sample variance 1 suggests the statistical process model

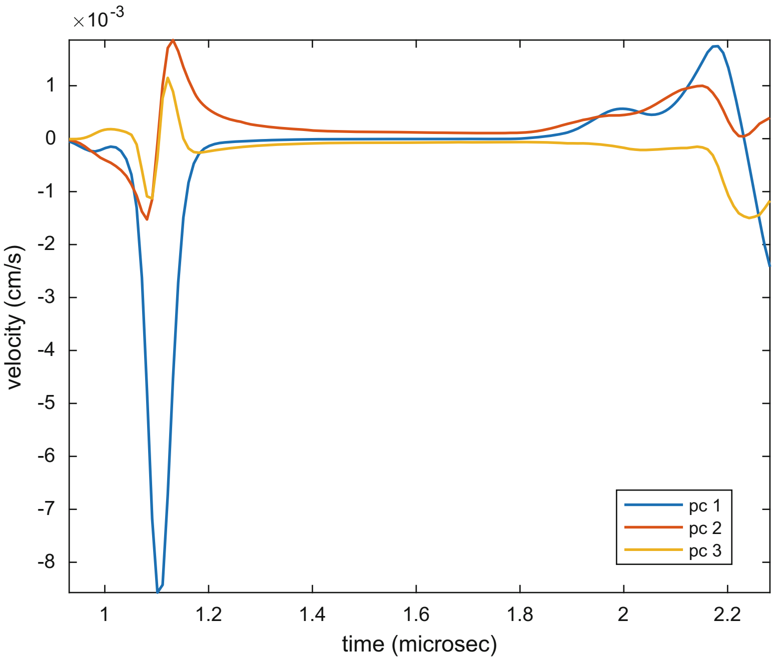


Fig. 8.15 Unscaled eigenvectors constituting the simulator basis model

$$\boldsymbol{Y}^{s,n}(\boldsymbol{x}^s,\boldsymbol{t}) = \boldsymbol{K}^s\,\boldsymbol{W}^s(\boldsymbol{x}^s,\boldsymbol{t}) \tag{8.4.5}$$

for the standardized $\boldsymbol{y}^{s,n}(\boldsymbol{x}^s,\boldsymbol{t})$ over the $(\boldsymbol{x}^s,\boldsymbol{t})$ input domain. Coefficient model $\boldsymbol{W}^s(\boldsymbol{x}^s,\boldsymbol{t}) = \left(W_1^s(\boldsymbol{x}^s,\boldsymbol{t}),\ldots,W_{p_s}^s(\boldsymbol{x}^s,\boldsymbol{t})\right)$ is assumed to have mutually independent components, each with mean zero. To complete the model, it is assumed that (marginally) each $W_i^s(\boldsymbol{x}^s,\boldsymbol{t})$, $i = 1,\ldots,p_s$, is a GP with precision $\lambda_{s,i}$ and *correlation* function $R_{s,i}(\cdot,\cdot)$ that is specified by (8.2.1). Each component process could be constrained to have variance 1 by setting $\lambda_{s,i} = 1$ for $i = 1,\ldots,p_s$; however, this restriction is relaxed as described in Sect. 8.5.1 to require these precisions to take values "near" 1.

In sum, the marginal process model is described in the notation (8.2.2) by

$$\left[W_i^s(\boldsymbol{x}^s,\boldsymbol{t})\,\middle|\,\lambda_{s,i},\boldsymbol{\rho}^{x,i},\boldsymbol{\rho}^{t,i}\right] \sim GP\left(0,\lambda_{s,i},(\boldsymbol{\rho}^{x,i},\boldsymbol{\rho}^{t,i})\right) \text{ for } i = 1,\ldots,p_s\,, \tag{8.4.6}$$

and jointly $\{W_i^s(\boldsymbol{x}^s,\boldsymbol{t})\}_{i=1}^{p_s}$ are assumed to have *covariance* matrix

$$\begin{bmatrix} (1/\lambda_{s,1})R_{s,1}(\cdot,\cdot) & 0 & \cdots & 0 \\ 0 & (1/\lambda_{s,2})R_{s,2}(\cdot,\cdot) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & (1/\lambda_{s,p_s})R_{s,p_s}(\cdot,\cdot) \end{bmatrix}.$$

There are two caveats to this basic formulation. First, for numerical stability, the diagonals of the covariance of the i-th coefficient process $W_i^s(\boldsymbol{x}^s, \boldsymbol{t})$ are augmented by an independent, mean-zero Gaussian *nugget effect process* having precision $\lambda_{n,i}$, resulting in the covariance function

$$Cov\left[W_i^s(\boldsymbol{x}_1, \boldsymbol{t}_1), W_i^s(\boldsymbol{x}_2, \boldsymbol{t}_2)\right] = \frac{1}{\lambda_{s,i}} R_{s,i}\left((\boldsymbol{x}_1, \boldsymbol{t}_1), (\boldsymbol{x}_2, \boldsymbol{t}_2)\right) + \frac{1}{\lambda_{n,i}} I\left\{(\boldsymbol{x}_1, \boldsymbol{t}_1) = (\boldsymbol{x}_2, \boldsymbol{t}_2)\right\}$$

for $i = 1, \ldots, p_s$, where $I\{\cdot\}$ is the indicator function which takes the value 1 if its argument is true, and 0 otherwise. The p_s nugget effect processes are assumed to be mutually independent.

Second, recall that the statistical model (8.4.5) excludes the $m_s - p_s$ basis vectors in $\boldsymbol{U}_2^s$ from the representation of standardized simulation output due to their insignificance in explaining variation in the observed simulation output. The error in excluding these basis vectors is termed the *simulator basis noise*. Similar to the nugget effect processes, the value of the simulator basis noise is described as a draw from a mean-zero multivariate normal random vector $\boldsymbol{\varepsilon}^s$ having covariance matrix $(1/\lambda_{s,\varepsilon})\boldsymbol{I}_{m_s}$, resulting in an adjusted statistical model of standardized simulation output:

$$\boldsymbol{Y}^{s,n,\varepsilon}(\boldsymbol{x}^s, \boldsymbol{t}) \equiv \boldsymbol{Y}^{s,n}(\boldsymbol{x}^s, \boldsymbol{t}) + \boldsymbol{\varepsilon}^s = \boldsymbol{K}^s\, \boldsymbol{W}^s(\boldsymbol{x}^s, \boldsymbol{t}) + \boldsymbol{\varepsilon}^s\,. \tag{8.4.7}$$

The multivariate normal distribution is parameterized as in (B.1.1) of Appendix B.1.

Example 1.7 *(Continued)* (*Case 1: Simulator Emulation*). The matrix of simulation runs $\boldsymbol{Y}^s$, having 128 columns, is partitioned into $m = 127$ simulation runs used to train an emulator and 1 hold-out simulation run used to test the emulator. Figure 8.16 shows the centered training and test sets, along with a least squares fit to the centered test run calculated assuming (8.4.7). Centering was accomplished by subtracting the mean vector of the 127 training runs. Although the restriction to the $p_s = 3$ dominant simulator basis vectors ($\boldsymbol{K}^s$) yields slight overprediction in the flat portion of the velocity profile, it appears that model (8.4.7) adequately captures the velocity behavior of the test run in the more variable regions of the time domain. ♦

Applying (8.4.7) to the design locations at which standardized simulation output is calculated yields

$$\boldsymbol{Y}^{s,n,\varepsilon}(\boldsymbol{x}_i^s, \boldsymbol{t}_i) = \boldsymbol{K}^s\, \boldsymbol{W}^s(\boldsymbol{x}_i^s, \boldsymbol{t}_i) + \boldsymbol{\varepsilon}_i^s\,,\ i = 1, \ldots, m\,, \tag{8.4.8}$$

where the simulator basis noise vectors $\{\boldsymbol{\varepsilon}_i^s\}_{i=1}^m$ are modeled as being mutually independent.

In the case where $p_s = m_s$, which is typical for multivariate applications, the simulator basis noise is extraneous, and model (8.4.5) will be assumed. Applying (8.4.5) to the design locations at which standardized simulation output is calculated yields

$$\boldsymbol{Y}^{s,n}(\boldsymbol{x}_i^s, \boldsymbol{t}_i) = \boldsymbol{K}^s\, \boldsymbol{W}^s(\boldsymbol{x}_i^s, \boldsymbol{t}_i)\,,\ i = 1, \ldots, m\,. \tag{8.4.9}$$

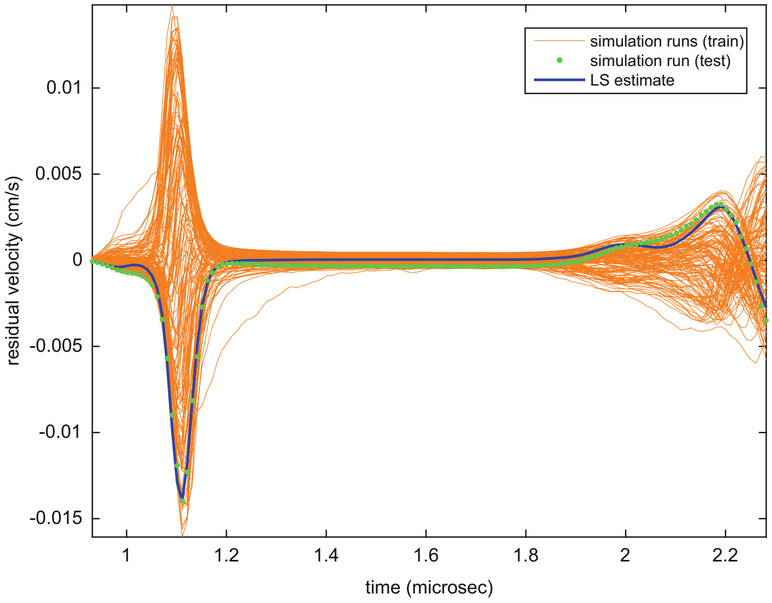


Fig. 8.16 One hundred and twenty-seven centered simulation training runs (orange lines), centered simulation test run (green dots), and the least squares fit to the centered simulation test run based on the three input variable regressions from $\boldsymbol{K}^s$ (blue line)

Statistical models (8.4.5) and (8.4.7) are often referred to as *emulator* models, as they provide a framework for representing simulation output and predicting it with quantification of uncertainty at arbitrary input settings in the input domain. This prediction capability will be described in Sect. 8.5.2.

Below, the collection of parameters requiring statistical inference for model (8.4.5) is denoted by

$$\boldsymbol{\Omega}^s = \left(\lambda_{s,1}, \ldots, \lambda_{s,p_s}, \lambda_{n,1}, \ldots, \lambda_{n,p_s}, \boldsymbol{\rho}^{x,1}, \boldsymbol{\rho}^{t,1}, \ldots, \boldsymbol{\rho}^{x,p_s}, \boldsymbol{\rho}^{t,p_s}\right). \tag{8.4.10}$$

This parameter set is augmented by $\lambda_{s,\varepsilon}$ for model (8.4.7): $\boldsymbol{\Omega}^{s,\varepsilon} = \{\boldsymbol{\Omega}^s, \lambda_{s,\varepsilon}\}$.

8.4.2 The Experimental Data

Let $\boldsymbol{y}^p(\boldsymbol{x}_1^p), \ldots, \boldsymbol{y}^p(\boldsymbol{x}_n^p)$ denote the outputs for n physical system experiments. Recall that the components of each $\boldsymbol{x}^p$ and $\boldsymbol{x}^s$ represent the *same* input variables; the superscripts are used to indicate that distinct settings of the input vectors are allowed when collecting physical observations versus simulation run outputs. Let $m_{p,i}$ denote the length of $\boldsymbol{y}^p(\boldsymbol{x}_i^p)$ for $i = 1, \ldots, n$. In typical functional data applications, the $\boldsymbol{y}^p(\boldsymbol{x}_i^p)$, $i = 1, \ldots, n$, are observed on a *less dense mesh* of index variable(s) settings than are the simulations so that $m_{p,i} \leq m_s$ for each experiment. On the other hand, applications with multivariate output are assumed to correspond to the same responses as produced by simulations, so that $m_{p,i} = m_s$ holds for all experiments.

The experimental output vector $\boldsymbol{y}^p(\boldsymbol{x}_i^p)$ is assumed to be realized from the following statistical model:

$$\boldsymbol{Y}^p(\boldsymbol{x}_i^p) = \boldsymbol{\mu}(\boldsymbol{x}_i^p) + \boldsymbol{\varepsilon}^p(\boldsymbol{x}_i^p)\,,\ i = 1, \ldots, n\,, \tag{8.4.11}$$

where $\boldsymbol{\mu}(\boldsymbol{x}_i^p)$ denotes the mean response of the physical system when the control variables are set at $\boldsymbol{x}_i^p$ and realized on the mesh of index variable(s) corresponding to the i-th experiment. The *observation error* for the i-th experiment involving the physical system, $\boldsymbol{\varepsilon}^p(\boldsymbol{x}_i^p)$, is assumed to be a mean-zero multivariate normal random vector having precision matrix $\lambda_{p,\varepsilon}\boldsymbol{P}_i^p$:

$$\boldsymbol{\varepsilon}^p(\boldsymbol{x}_i^p) \sim N_{m_{p,i}}\left(\boldsymbol{0}_{m_{p,i}}, \frac{1}{\lambda_{p,\varepsilon}}\left(\boldsymbol{P}_i^p\right)^{-1}\right)\,,\ i = 1, \ldots, n\,, \tag{8.4.12}$$

where $\boldsymbol{0}_r$ is an $r \times 1$ vector of zeros. The $m_{p,i} \times m_{p,i}$ positive definite matrix $\boldsymbol{P}_i^p$ is assumed fixed for $i = 1, \ldots, n$, and the error terms $\boldsymbol{\varepsilon}^p(\cdot)$ are mutually independent across the n experiments. The formulation (8.4.11) represents the extension of (8.2.4) to the functional data setting.

Analogous to the emulator development in Sect. 8.4.1, the experimental data vectors are centered and scaled but using the summary statistics from the *simulation output*. In the *functional case*, $m_s \gg m_{p,i}$, the mean simulation output vector $\overline{\boldsymbol{y}}^s = (1/m)\boldsymbol{Y}^s\boldsymbol{1}_m$ is interpolated onto the index variable(s) mesh corresponding to the i-th experiment to produce the $m_{p,i} \times 1$ vector $\overline{\boldsymbol{y}}_i^s$ for $i = 1, \ldots, n$. The i-th experimental output vector is centered by subtracting this interpolated simulation mean:

$$\boldsymbol{y}^{p,c}(\boldsymbol{x}_i^p) = \boldsymbol{y}^p(\boldsymbol{x}_i^p) - \overline{\boldsymbol{y}}_i^s\,,\ i = 1, \ldots, n\,.$$

The standardized output from the i-th experiment is then obtained as

$$\boldsymbol{y}^{p,n}(\boldsymbol{x}_i^p) = (1/\varsigma^s)\,\boldsymbol{y}^{p,c}(\boldsymbol{x}_i^p)\,,\ i = 1, \ldots, n\,,$$

with ς^s given by (8.4.2). In the *multivariate case*, $m_{p,i} = m_s$ for $i = 1, \ldots, n$, the standardized output from the i-th experiment is given by

$$\boldsymbol{y}^{p,n}(\boldsymbol{x}_i^p) = \left(\boldsymbol{C}_d^s\right)^{-1/2}\left(\boldsymbol{y}^p(\boldsymbol{x}_i^p) - \overline{\boldsymbol{y}}^s\right)\,, i = 1, \ldots, n\,,$$

with $\boldsymbol{C}_d^s$ given by (8.4.3).

Adapting (8.4.11), the standardized experimental output $\boldsymbol{y}^{p,n}(\boldsymbol{x}_i^p)$, $i = 1, \ldots, n$, is therefore assumed to be a realization from the statistical model:

$$\boldsymbol{Y}^{p,n}(\boldsymbol{x}_i^p) = \boldsymbol{\mu}^n(\boldsymbol{x}_i^p) + \boldsymbol{\varepsilon}^{p,n}(\boldsymbol{x}_i^p)\,, \tag{8.4.13}$$

where $\boldsymbol{Y}^{p,n}(\boldsymbol{x}_i^p) = (1/\varsigma^s)\left(\boldsymbol{Y}^p(\boldsymbol{x}_i^p) - \overline{\boldsymbol{y}}_i^s\right)$ is a vector of length $m_{p,i}$

$$\boldsymbol{\mu}^n(\boldsymbol{x}_i^p) = (1/\varsigma^s)\left(\boldsymbol{\mu}(\boldsymbol{x}_i^p) - \overline{\boldsymbol{y}}_i^s\right), \text{ and}$$
$$\boldsymbol{\varepsilon}^{p,n}(\boldsymbol{x}_i^p) = (1/\varsigma^s)\,\boldsymbol{\varepsilon}^p(\boldsymbol{x}_i^p)$$

in the functional case, while $\boldsymbol{Y}^{p,n}(\boldsymbol{x}_i^p) = (\boldsymbol{C}_d^s)^{-1/2}(\boldsymbol{Y}^p(\boldsymbol{x}_i^p) - \overline{\boldsymbol{y}}^s)$ is a vector of length m_s

$$\boldsymbol{\mu}^n(\boldsymbol{x}_i^p) = \left(\boldsymbol{C}_d^s\right)^{-1/2}\left(\boldsymbol{\mu}(\boldsymbol{x}_i^p) - \overline{\boldsymbol{y}}^s\right), \text{ and}$$
$$\boldsymbol{\varepsilon}^{p,n}(\boldsymbol{x}_i^p) = \left(\boldsymbol{C}_d^s\right)^{-1/2}\boldsymbol{\varepsilon}^p(\boldsymbol{x}_i^p)$$

in the multivariate case. Hence setting

$$\boldsymbol{P}_i^{p,n} = (\varsigma^s)^2\,\boldsymbol{P}_i^p$$

in the functional case and

$$\boldsymbol{P}_i^{p,n} = \left(\boldsymbol{C}_d^s\right)^{1/2}\boldsymbol{P}_i^p\left(\boldsymbol{C}_d^s\right)^{1/2}$$

in the multivariate case gives

$$\boldsymbol{\varepsilon}^{p,n}(\boldsymbol{x}_i^p) \sim N_{m_{p,i}}\left(\boldsymbol{0}_{m_{p,i}}, \frac{1}{\lambda_{p,\varepsilon}}\left(\boldsymbol{P}_i^{p,n}\right)^{-1}\right) \tag{8.4.14}$$

for $i = 1, \ldots, n$, under the assumption (8.4.12).

Because $\boldsymbol{\mu}(\cdot)$ (and therefore $\boldsymbol{\mu}^n(\cdot)$) is not directly observable, models (8.4.11) and (8.4.13) are of limited use. However, recognizing that the simulator was built to provide a high-fidelity representation of the physical system, the simulation output can be used to build a statistical model of $\boldsymbol{\mu}^n(\cdot)$.

As in Sect. 8.2, letting $\boldsymbol{\theta}$ denote the unknown true value of the calibration parameter $\boldsymbol{t}$ in the physical system, the bias (discrepancy) of the simulator is defined analogously to (8.2.5) by

$$\boldsymbol{\delta}(\boldsymbol{x}^p) \equiv \boldsymbol{\mu}(\boldsymbol{x}^p) - \boldsymbol{y}^s(\boldsymbol{x}^p, \boldsymbol{\theta}),$$

where the index variable(s) mesh is assumed to contain m_p elements. The standardized version of simulator bias is defined as follows:

$$\boldsymbol{\delta}^n(\boldsymbol{x}^p) \equiv \boldsymbol{\mu}^n(\boldsymbol{x}^p) - \boldsymbol{y}^{s,n}(\boldsymbol{x}^p, \boldsymbol{\theta}), \tag{8.4.15}$$

where $\boldsymbol{\delta}^n(\boldsymbol{x}^p) = (1/\varsigma^s)\,\boldsymbol{\delta}(\boldsymbol{x}^p)$ and $\boldsymbol{\delta}^n(\boldsymbol{x}^p) = (\boldsymbol{C}_d^s)^{-1/2}\boldsymbol{\delta}(\boldsymbol{x}^p)$ in the functional and multivariate cases, respectively.

The discrepancy $\boldsymbol{\delta}^n(\boldsymbol{x}^p)$ is not directly observable. Thus it is modeled statistically as a realization of

$$\boldsymbol{\varDelta}^n(\boldsymbol{x}^p) = \boldsymbol{D}^p\,\boldsymbol{V}^p(\boldsymbol{x}^p), \tag{8.4.16}$$

where the $m_p \times p_\delta$ matrix $\boldsymbol{D}^p$ contains a fixed linearly independent set of user-selected basis vectors and $\boldsymbol{V}^p(\boldsymbol{x}^p)$ is a GP.

As examples of $\boldsymbol{D}^p$ in (8.4.16), in functional cases, $\boldsymbol{D}^p$ often takes the form of a kernel regression model. The index variable(s) domain is populated with p_δ kernel centers, and each column of $\boldsymbol{D}^p$ is determined by evaluating the kernel function on the index variable(s) mesh at each kernel center. In the multivariate case $m_p = m_s$, it is typical to choose $p_\delta = m_s$ and $\boldsymbol{D}^p = \boldsymbol{I}_{m_s}$, which represents the scenario in which each scalar output is associated with its own discrepancy correction.

Example 1.7 (Continued). Figure 8.17 shows $p_\delta = 9$ unscaled kernel basis vectors (columns of $\varsigma^s \boldsymbol{D}^p$). The kernels were researcher selected to be normal density functions with different center points and common standard deviation. Figure 8.17 plots the kernels at the $m_s = 136$ mesh of time values over [0.9314 μs, 2.2816 μs], the time interval used by the simulator runs (the one-dimensional index variable for these data). Specifically, seven of the kernels, plotted with blue lines, had equally spaced centers all contained within the domain of observed time points beginning at 0.9314 μs, ending at 2.2816 μs, and spaced at 0.225 μs. The remaining two kernels, plotted with orange lines, were centered 0.225 μs below and above the lower and upper boundaries of the time domain, respectively. These last two kernels provide a buffer to assist with the adequate representation of discrepancy functions in the vicinity of the boundaries. All nine kernels had common standard deviation equal to the kernel center separation distance of 0.225 μs. The columns of $\boldsymbol{D}^p$ were normalized for numerical stability by ensuring the maximal element of the matrix product $\boldsymbol{D}^p\,(\boldsymbol{D}^p)^\top$ is 1. ♦

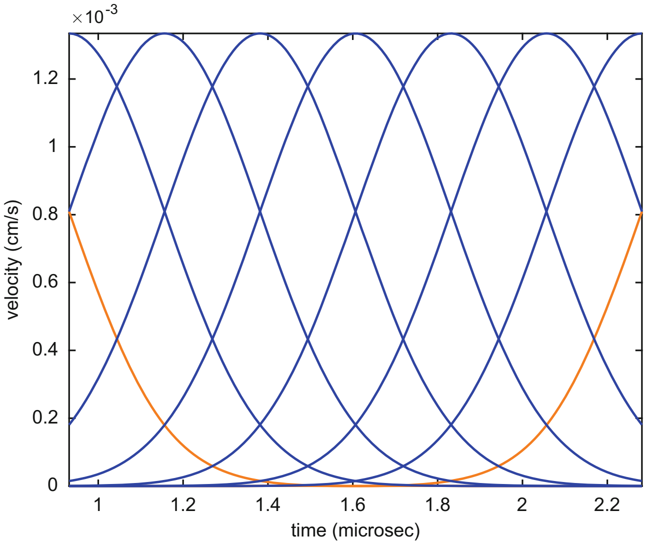


Fig. 8.17 Unscaled kernel basis vectors $\varsigma^s \boldsymbol{D}^p$ constituting the discrepancy basis model

An example of $\boldsymbol{V}^p(\boldsymbol{x}^p)$ in (8.4.16), and the process assumed in this section, can be motivated by applications in which the columns of $\boldsymbol{D}^p$ are partitioned into F groups

$G_1, \ldots, G_F$, where $p_\delta = \sum_{i=1}^F |G_i|$ with $|G_i|$ denoting the size of group G_i. This would be the situation in a functional case where each group represents a set of basis vectors in a regression model of the discrepancy for *distinct subregions* of index variable(s) space. Let $\boldsymbol{V}_i^p(\boldsymbol{x}^p)$ denote those coefficients of $\boldsymbol{V}^p(\boldsymbol{x}^p)$ that correspond to G_i for $i = 1, \ldots, F$. This section assumes the component processes $\{\boldsymbol{V}_i^p(\boldsymbol{x}^p)\}_{i=1}^F$ of $\boldsymbol{V}^p(\boldsymbol{x}^p)$ are mutually independent. In addition, it assumes that the individual processes $\{V_{i,j}^p(\boldsymbol{x}^p)\}_{j=1}^{|G_i|}$ constituting $\boldsymbol{V}_i^p(\boldsymbol{x}^p)$ can be modeled as mutually independent GPs with the (i, j) process having mean zero, precision $\lambda_{\delta,i}$, and correlation function $R_{\delta,i}(\cdot,\cdot)$ given by (8.2.6). In our earlier notation

$$\left[V_{i,j}^p(\boldsymbol{x}^p) \,\middle|\, \lambda_{\delta,i}, \boldsymbol{\rho}^{\delta,i}\right] \sim GP\left(0, \lambda_{\delta,i}, \boldsymbol{\rho}^{\delta,i}\right) \text{ for } j = 1, \ldots, |G_i|\,. \tag{8.4.17}$$

Below, the collection of parameters in the discrepancy model (8.4.16) requiring statistical inference is denoted by

$$\boldsymbol{\Omega}^\delta = \left(\lambda_{\delta,1}, \ldots, \lambda_{\delta,F}, \boldsymbol{\rho}^{\delta,1}, \ldots, \boldsymbol{\rho}^{\delta,F}\right).$$

In the functional case, $F = 1$ is often chosen to limit the number of discrepancy parameters in $\boldsymbol{\Omega}^\delta$ to a manageable size, as the extreme choice of $F = p_\delta$ when p_δ is large results in a large number of discrepancy parameters required to adequately model the simulator bias. In the multivariate case, $F = m_s$ is typically selected to equip the discrepancy process for each scalar output with its own set of discrepancy parameters. This does not result in an explosion of parameters, as the multivariate case generally involves small to moderate output dimension m_s.

Combining statistical models (8.4.5) and (8.4.16) and assuming independence between the signal, $\boldsymbol{W}^s(\cdot)$, and discrepancy, $\boldsymbol{V}^p(\cdot)$, processes, the definition of simulator bias (8.4.15) suggests the statistical model

$$\boldsymbol{M}^n(\boldsymbol{x}^p) = \boldsymbol{K}^s\, \boldsymbol{W}^s(\boldsymbol{x}^p, \boldsymbol{\Theta}) + \boldsymbol{D}^p\, \boldsymbol{V}^p(\boldsymbol{x}^p)$$

for $\boldsymbol{\mu}^n(\boldsymbol{x}^p)$ at arbitrary control input $\boldsymbol{x}^p$ contained within the input domain. Here a capital $\boldsymbol{\Theta}$ is used in place of the lowercase $\boldsymbol{\theta}$ to indicate the unknown true calibration parameter value that will be inferred statistically. A common input variable(s) mesh is assumed for each term in this model.

The formulation of the previous paragraph allows for an extension of (8.4.13) to a statistical model of the standardized output from the i-th experiment involving the emulator model:

$$\begin{aligned} \boldsymbol{Y}^{p,n}(\boldsymbol{x}_i^p) &= \boldsymbol{M}^n(\boldsymbol{x}_i^p) + \boldsymbol{\varepsilon}^{p,n}(\boldsymbol{x}_i^p) \\ &= \boldsymbol{K}_i^s\, \boldsymbol{W}^s(\boldsymbol{x}_i^p, \boldsymbol{\Theta}) + \boldsymbol{D}_i^p\, \boldsymbol{V}^p(\boldsymbol{x}_i^p) + \boldsymbol{\varepsilon}^{p,n}(\boldsymbol{x}_i^p)\,, \end{aligned} \tag{8.4.18}$$

for $i = 1, \ldots, n$. Here the processes $\{\boldsymbol{W}^s(\cdot), \boldsymbol{V}^p(\cdot), \boldsymbol{\varepsilon}^{p,n}(\cdot)\}$ are assumed mutually independent and independent across experiments. In the functional case, each column of the $\boldsymbol{K}^s$ matrix in (8.4.4) defined on the index variable(s) mesh from the simulation output must be interpolated onto the index variable(s) mesh corresponding to the i-th physical experiment to produce the $\boldsymbol{K}_i^s$ matrix. The columns of the $\boldsymbol{D}_i^p$ matrix arise from evaluating the kernel function on the input variable(s) mesh corresponding to the i-th experiment at each kernel center. In the multivariate case, $\boldsymbol{K}_i^s = \boldsymbol{K}^s$ with $\boldsymbol{K}^s$ given by (8.4.4) for $i = 1, \ldots, n$. Furthermore, each $\boldsymbol{D}_i^p$ is set to a common matrix of basis vectors $\boldsymbol{D}^p$ for $i = 1, \ldots, n$, typically $\boldsymbol{D}^p = \boldsymbol{I}_{m_s}$.

Example 1.7 *(Continued)* (*Case 2: Emulation of the Calibrated Simulator Output Allowing for Simulator Bias*). Figure 8.18 shows the centered data $\boldsymbol{y}^{p,c}(\boldsymbol{x}_1^p)$ for the $n = 1$ experiment available for calibration, superimposed on the $m = 128$ centered simulation runs constituting the columns of $\boldsymbol{Y}^{s,c}$, and a least squares fit to $\boldsymbol{y}^{p,c}(\boldsymbol{x}_1^p)$ calculated assuming (8.4.18).

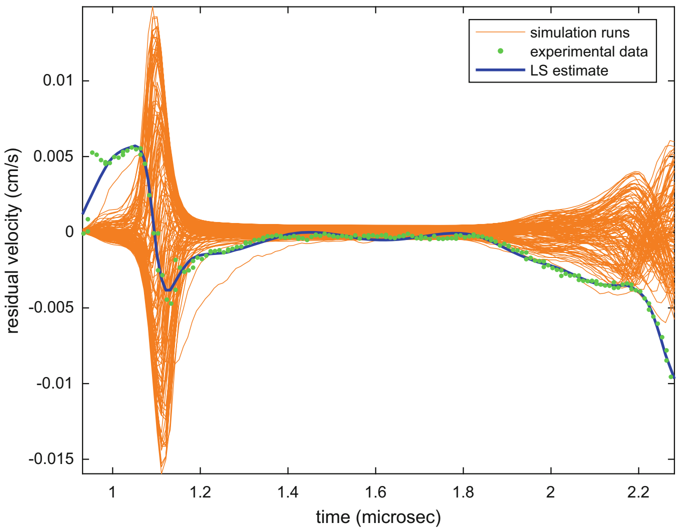


Fig. 8.18 One hundred and twenty-eight centered simulation runs (orange lines), centered experimental data (green dots), and the least squares fit to the centered experimental data based on the 12 (3 + 9) input variable regression from $\boldsymbol{K}_1^s$ and $\boldsymbol{D}_1^p$ (blue line)

The matrices $\boldsymbol{K}_1^s$ and $\boldsymbol{D}_1^p$ of (8.4.18) were obtained by linearly interpolating each column of $\boldsymbol{K}^s$ and $\boldsymbol{D}^p$ onto the $m_{p,1} = 170$ time points corresponding to the experimental output (a rare instance in which the index variable mesh for the experimental output is denser than that used by the simulator). The precision matrix $\boldsymbol{P}_1^{p,n}$ of the error process in Eq. (8.4.14) is taken to be $\boldsymbol{I}_{170}$. It is apparent that model (8.4.18), composed of three simulator basis vectors ($\boldsymbol{K}_1^s$) and nine discrepancy basis vectors ($\boldsymbol{D}_1^p$), adequately represents the variation in the experimental data with the exception of the observed sharp velocity increase in early time. If necessary, an improved representation could be obtained by placing additional kernel basis vectors in this region. ♦

Below, the complete set of parameters in (8.4.18) requiring statistical inference is denoted by

$$\boldsymbol{\Omega}^{p,\mathrm{d},\varepsilon} = \{\boldsymbol{\Omega}^{p,\mathrm{d}}, \lambda_{p,\varepsilon}\}\,, \tag{8.4.19}$$

where $\boldsymbol{\Omega}^{p,\mathrm{d}} = \{\boldsymbol{\Theta}, \boldsymbol{\Omega}^{s}, \boldsymbol{\Omega}^{\delta}\}$. In the scenario for which it is assumed that $\boldsymbol{\delta}(\boldsymbol{x}^p) \equiv \boldsymbol{0}$ for every $\boldsymbol{x}^p$ in the input domain, the statistical model (8.4.18) reduces to

$$\begin{aligned} \boldsymbol{Y}^{p,n}(\boldsymbol{x}_i^p) &= \boldsymbol{M}_0^n(\boldsymbol{x}_i^p) + \boldsymbol{\varepsilon}^{p,n}(\boldsymbol{x}_i^p) \\ &= \boldsymbol{K}_i^s\, \boldsymbol{W}^s(\boldsymbol{x}_i^p, \boldsymbol{\Theta}) + \boldsymbol{\varepsilon}^{p,n}(\boldsymbol{x}_i^p)\,,\ i = 1, \ldots, n\,. \end{aligned} \tag{8.4.20}$$

Example 1.7 (Continued) (Case 3: Emulation of the Calibrated Simulator Output Assuming No Simulator Bias). Figure 8.19 shows centered data $\boldsymbol{y}^{p,c}(\boldsymbol{x}_1^p)$ for the $n = 1$ experiment available for calibration, superimposed on the $m = 128$ centered simulation runs constituting the columns of $\boldsymbol{Y}^{s,c}$, and a three variable least squares fit to $\boldsymbol{y}^{p,c}(\boldsymbol{x}_1^p)$ calculated assuming (8.4.20).

The matrix $\boldsymbol{K}_1^s$ of (8.4.20) was obtained by linearly interpolating each column of $\boldsymbol{K}^s$ onto the $m_{p,1} = 170$ time points corresponding to experimental output. The precision matrix $\boldsymbol{P}_1^{p,n}$ of the error process (8.4.14) is taken to be $\boldsymbol{I}_{170}$. It is apparent that model (8.4.20), composed of three simulator basis vectors ($\boldsymbol{K}_1^s$) and no discrepancy component, is able to capture the behavior in the experimental data reasonably well between approximately 1.1 and 2.2 µs, but has considerable discrepancy at early and late times. ♦

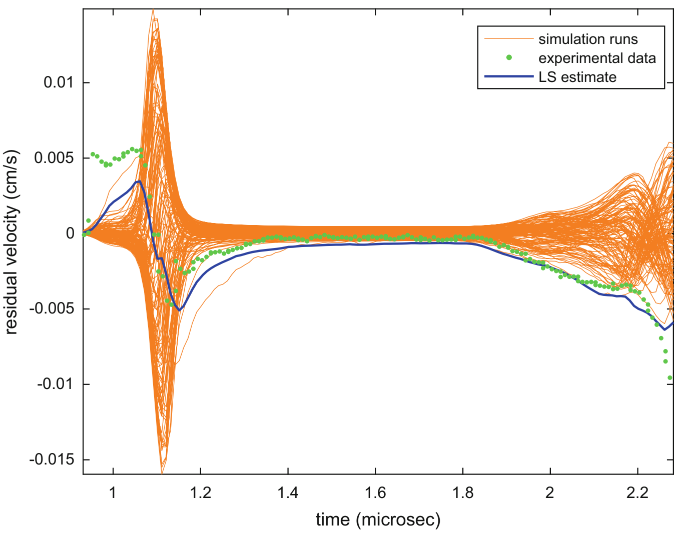


Fig. 8.19 One hundred and twenty-eight centered simulation runs (orange lines), centered experimental data (green dots), and the least squares fit to the centered experimental data based on the three input variable regression from $\boldsymbol{K}_1^s$ (blue line)

The complete set of parameters in (8.4.20) requiring statistical inference is denoted by

$$\boldsymbol{\Omega}^{p,\,\text{nod},\varepsilon} = \{\boldsymbol{\Omega}^{p,\,\text{nod}}, \lambda_{p,\varepsilon}\}\,, \tag{8.4.21}$$

where $\boldsymbol{\Omega}^{p,\,\text{nod}} = \{\boldsymbol{\Theta}, \boldsymbol{\Omega}^s\}$.

8.4.3 Joint Statistical Models and Log Likelihood Functions

The goal of this section is to derive the joint statistical models that are used in the Bayesian analysis presented in Sect. 8.5.1. The joint models combine (8.4.7) for the standardized simulation output in Sect. 8.4.1 which includes the simulator basis noise and those for the standardized physical experiment output in Sect. 8.4.2 both with and without simulator discrepancy. In the derivation below, recall that p_s denotes the number of mutually independent $W_i^s(\cdot,\cdot)$, $i = 1,\ldots,p_s$, processes that are used to describe the simulation output in (8.4.8) and the experimental mean in (8.4.18) or (8.4.20). Also p_δ denotes the number of discrepancy basis functions where $p_\delta = \sum_{i=1}^F |G_i|$ for the model that uses separately parameterized discrepancies for F distinct subregions of the index variable space. For the result-oriented reader, the log likelihood functions for the models with and without simulator basis error are (8.4.51) and (8.4.67), respectively.

8.4.3.1 Joint Statistical Model That Allows Simulator Discrepancy

Let the $m \times p_s$ matrix $\boldsymbol{W}_s$ collect the basis coefficient processes in the emulator model,

$$\boldsymbol{W}_s = \left[\boldsymbol{W}^s(\boldsymbol{x}_1^s, \boldsymbol{t}_1)\ \boldsymbol{W}^s(\boldsymbol{x}_2^s, \boldsymbol{t}_2)\ \cdots\ \boldsymbol{W}^s(\boldsymbol{x}_m^s, \boldsymbol{t}_m)\right]^\top\,,$$

while the $n \times p_s$ matrix

$$\boldsymbol{U}_s = \left[\boldsymbol{W}^s(\boldsymbol{x}_1^p, \boldsymbol{\Theta})\ \boldsymbol{W}^s(\boldsymbol{x}_2^p, \boldsymbol{\Theta})\ \cdots\ \boldsymbol{W}^s(\boldsymbol{x}_n^p, \boldsymbol{\Theta})\right]^\top,$$

and the $n \times p_\delta$ matrix

$$\boldsymbol{V}_p = \left[\boldsymbol{V}^p(\boldsymbol{x}_1^p)\ \boldsymbol{V}^p(\boldsymbol{x}_2^p)\cdots\ \boldsymbol{V}^p(\boldsymbol{x}_n^p)\right]^\top\,,$$

denote the basis coefficient processes in the calibrated emulator model of the experimental data mean and the discrepancy of the calibrated simulator mean from the true mean, respectively (see Sects. 8.4.1 and 8.4.2).

When the simulator discrepancy is modeled statistically as in (8.4.18), the joint statistical model will be specified in terms of the vector $\boldsymbol{Z}_\text{d}$:

$$\boldsymbol{Z}_\text{d} = \begin{pmatrix} \text{Vec}\left(\left[\boldsymbol{V}_p\ \boldsymbol{U}_s\right]\right) \\ \text{Vec}\,(\boldsymbol{W}_s) \end{pmatrix} = \begin{pmatrix} \text{Vec}\left(\boldsymbol{V}_p\right) \\ \text{Vec}\,(\boldsymbol{U}_s) \\ \text{Vec}\,(\boldsymbol{W}_s) \end{pmatrix} \tag{8.4.22}$$

of length $n(p_\delta + p_s) + mp_s$, where the Vec$(\cdot)$ operator is the column vector obtained by stacking the column vectors of its matrix argument in order from left to right (see Appendix B.5). In words, $\boldsymbol{Z}_\text{d}$ first stacks $V_1^p(\boldsymbol{x}_1^p), \ldots, V_1^p(\boldsymbol{x}_n^p)$, followed by the n $V_2^p(\cdot)$ values, followed by the $V_3^p(\cdot)$ through $V_{p_\delta}^p(\cdot)$ values. The $\boldsymbol{U}_s$ and $\boldsymbol{W}_s$ matrices are analogously stacked by grouping all coefficients in each of the p_s columns of $\boldsymbol{U}_s$ and finally all coefficients in each of the p_s columns of $\boldsymbol{W}_s$.

Because $\boldsymbol{Z}_\text{d}$ is constructed from mean-zero GPs, conditional on $\boldsymbol{\Omega}^{p,\text{d}}$, it is multivariate normally distributed with zero mean vector and covariance matrix:

$$\boldsymbol{\Sigma}_{Z,\text{d}} = \begin{pmatrix} \boldsymbol{\Sigma}_V & \boldsymbol{0}_{np_\delta,np_s} & \boldsymbol{0}_{np_\delta,mp_s} \\ \boldsymbol{0}_{np_s,np_\delta} & \boldsymbol{\Sigma}_U & \boldsymbol{\Sigma}_{UW} \\ \boldsymbol{0}_{mp_s,np_\delta} & \boldsymbol{\Sigma}_{UW}^\top & \boldsymbol{\Sigma}_W \end{pmatrix}, \tag{8.4.23}$$

where $\boldsymbol{0}_{r,s}$ is the $r \times s$ matrix of zeros. The block matrix components of $\boldsymbol{\Sigma}_{Z,\text{d}}$ themselves have block diagonal structures:

$$\begin{aligned} \boldsymbol{\Sigma}_V &= diag(\boldsymbol{\Sigma}_{V,1}, \boldsymbol{\Sigma}_{V,2}, \ldots, \boldsymbol{\Sigma}_{V,p_\delta}), \\ \boldsymbol{\Sigma}_U &= diag(\boldsymbol{\Sigma}_{U,1}, \boldsymbol{\Sigma}_{U,2}, \ldots, \boldsymbol{\Sigma}_{U,p_s}), \\ \boldsymbol{\Sigma}_W &= diag(\boldsymbol{\Sigma}_{W,1}, \boldsymbol{\Sigma}_{W,2}, \ldots, \boldsymbol{\Sigma}_{W,p_s}), \text{ and} \qquad (8.4.24) \\ \boldsymbol{\Sigma}_{UW} &= diag(\boldsymbol{\Sigma}_{UW,1}, \boldsymbol{\Sigma}_{UW,2}, \ldots, \boldsymbol{\Sigma}_{UW,p_s}). \qquad (8.4.25) \end{aligned}$$

For $k \in \{1, \ldots, p_\delta\}$ and $i, j = 1, \ldots, n$, the (i, j) element of $\boldsymbol{\Sigma}_{V,k}$ is

$$R_{\delta,l}(\boldsymbol{x}_i^p, \boldsymbol{x}_j^p)/\lambda_{\delta,l},$$

where $l \in \{1, \ldots, F\}$ is chosen so that $k \in G_l$. For $k \in \{1, \ldots, p_s\}$ and $i, j = 1, \ldots, n$, the (i, j) element of $\boldsymbol{\Sigma}_{U,k}$ is

$$\frac{1}{\lambda_{s,k}} R_{s,k}\left((\boldsymbol{x}_i^p, \boldsymbol{\Theta}), (\boldsymbol{x}_j^p, \boldsymbol{\Theta})\right) + \frac{1}{\lambda_{n,k}} I\{\boldsymbol{x}_i^p = \boldsymbol{x}_j^p\}.$$

For $k \in \{1, \ldots, p_s\}$ and $i, j = 1, \ldots, m$, the (i, j) element of $\boldsymbol{\Sigma}_{W,k}$ is

$$\frac{1}{\lambda_{s,k}} R_{s,k}\left((\boldsymbol{x}_i^s, \boldsymbol{t}_i), (\boldsymbol{x}_j^s, \boldsymbol{t}_j)\right) + \frac{1}{\lambda_{n,k}} I\left\{(\boldsymbol{x}_i^s, \boldsymbol{t}_i) = (\boldsymbol{x}_j^s, \boldsymbol{t}_j)\right\}.$$

Finally, for $k \in \{1, \ldots, p_s\}$, $i = 1, \ldots, n$, and $j = 1, \ldots, m$, the (i, j) element of $\boldsymbol{\Sigma}_{UW,k}$ is

$$R_{s,k}\left((\boldsymbol{x}_i^p, \boldsymbol{\Theta}), (\boldsymbol{x}_j^s, \boldsymbol{t}_j)\right)/\lambda_{s,k}.$$

The vectors $\boldsymbol{Y}_{p,n}$ and $\boldsymbol{E}_{p,n}$ of length $m_y = \sum_{i=1}^n m_{p,i}$ and the vectors $\boldsymbol{Y}_{s,n,\varepsilon}$ and $\boldsymbol{E}_s$ of length $m_s m$ collect the standardized physical experiment output, observation errors, standardized simulation output, and simulator basis noise elements:

$$\boldsymbol{Y}_{p,n} = \left[\boldsymbol{Y}^{p,n}(\boldsymbol{x}_1^p)^\top \; \boldsymbol{Y}^{p,n}(\boldsymbol{x}_2^p)^\top \cdots \boldsymbol{Y}^{p,n}(\boldsymbol{x}_n^p)^\top\right]^\top,$$

$$\begin{aligned}
\boldsymbol{E}_{p,n} &= \left[\boldsymbol{\varepsilon}^{p,n}(\boldsymbol{x}_1^p)^\top \ \boldsymbol{\varepsilon}^{p,n}(\boldsymbol{x}_2^p)^\top \cdots \boldsymbol{\varepsilon}^{p,n}(\boldsymbol{x}_n^p)^\top\right]^\top , \\
\boldsymbol{Y}_{s,n,\varepsilon} &= \mathrm{Vec}\left(\left[\boldsymbol{Y}^{s,n,\varepsilon}(\boldsymbol{x}_1^s, \boldsymbol{t}_1) \ \boldsymbol{Y}^{s,n,\varepsilon}(\boldsymbol{x}_2^s, \boldsymbol{t}_2) \cdots \boldsymbol{Y}^{s,n,\varepsilon}(\boldsymbol{x}_m^s, \boldsymbol{t}_m)\right]\right) , \\
\boldsymbol{E}_s &= \mathrm{Vec}\left(\left[\boldsymbol{\varepsilon}_1^s \ \boldsymbol{\varepsilon}_2^s \cdots \boldsymbol{\varepsilon}_m^s\right]\right) ,
\end{aligned}$$

respectively. Using the above notation, the joint statistical model takes the following form:

$$\begin{pmatrix} \boldsymbol{Y}_{p,n} \\ \boldsymbol{Y}_{s,n,\varepsilon} \end{pmatrix} = \begin{pmatrix} \boldsymbol{X}_{p,\mathrm{d}} & \boldsymbol{0}_{m_y,mp_s} \\ \boldsymbol{0}_{m_sm,n(p_\delta+p_s)} & \boldsymbol{X}_s \end{pmatrix} \boldsymbol{Z}_\mathrm{d} + \begin{pmatrix} \boldsymbol{E}_{p,n} \\ \boldsymbol{E}_s \end{pmatrix}, \tag{8.4.26}$$

where $\boldsymbol{X}_{p,\mathrm{d}}$ and $\boldsymbol{X}_s$ are defined implicitly by (8.4.18) and (8.4.8), respectively. From the discussion preceeding (8.4.23), conditionally

$$\left[\boldsymbol{Z}_\mathrm{d} \,\middle|\, \boldsymbol{\Omega}^{p,\mathrm{d}}\right] \sim N_{n(p_\delta+p_s)+mp_s}\left(\boldsymbol{0}_{n(p_\delta+p_s)+mp_s}, \boldsymbol{\Sigma}_{Z,\mathrm{d}}\right) .$$

Utilizing (8.4.14), the discussion above (8.4.7), and the assumed mutual independence between simulator basis noise and observation error gives

$$\left[\begin{pmatrix} \boldsymbol{E}_{p,n} \\ \boldsymbol{E}_s \end{pmatrix} \middle| \lambda_{p,\varepsilon}, \lambda_{s,\varepsilon}\right] \sim N_{m_y+m_sm}\left(\boldsymbol{0}_{m_y+m_sm}, \boldsymbol{\Sigma}_E\right)$$

independent of $\boldsymbol{Z}_\mathrm{d}$, where

$$\boldsymbol{\Sigma}_E = \begin{pmatrix} \lambda_{p,\varepsilon}^{-1} (\boldsymbol{P}^{p,n})^{-1} & \boldsymbol{0}_{m_y,m_sm} \\ \boldsymbol{0}_{m_sm,m_y} & \lambda_{s,\varepsilon}^{-1} \boldsymbol{I}_{m_sm} \end{pmatrix} \tag{8.4.27}$$

with

$$\boldsymbol{P}^{p,n} = diag(\boldsymbol{P}_1^{p,n}, \boldsymbol{P}_2^{p,n}, \ldots, \boldsymbol{P}_n^{p,n}) .$$

Collecting the regression matrices of (8.4.18) into a $m_y \times n(p_\delta + p_s)$ block diagonal matrix,

$$\boldsymbol{F}_{DK} = diag\left(\left[\boldsymbol{D}_1^p \ \boldsymbol{K}_1^s\right], \left[\boldsymbol{D}_2^p \ \boldsymbol{K}_2^s\right], \ldots, \left[\boldsymbol{D}_n^p \ \boldsymbol{K}_n^s\right]\right) ,$$

and letting the $n(p_\delta + p_s)$ by $n(p_\delta + p_s)$ matrix $\boldsymbol{Q}_{n,p_\delta+p_s}$ denote a vec-permutation matrix as defined in Appendix B.5, $\boldsymbol{X}_{p,\mathrm{d}}$ is given by

$$\boldsymbol{X}_{p,\mathrm{d}} = \boldsymbol{F}_{DK}\, \boldsymbol{Q}_{n,p_\delta+p_s} . \tag{8.4.28}$$

This is seen by using the partitioning of $\boldsymbol{Z}_\mathrm{d}$ given in (8.4.22), noting that

$$\boldsymbol{Q}_{n,p_\delta+p_s} \mathrm{Vec}\left(\left[\boldsymbol{V}_p \ \boldsymbol{U}_s\right]\right) = \mathrm{Vec}\left(\begin{bmatrix} \boldsymbol{V}_p^\top \\ \boldsymbol{U}_s^\top \end{bmatrix}\right)$$

via (B.5.4) of Appendix B.5 and finally observing that

$$\boldsymbol{F}_{DK}\text{Vec}\left(\begin{bmatrix}\boldsymbol{V}_p^\top\\ \boldsymbol{U}_s^\top\end{bmatrix}\right) = \left[\boldsymbol{M}^n(\boldsymbol{x}_1^p)^\top\ \boldsymbol{M}^n(\boldsymbol{x}_2^p)^\top\ \cdots\ \boldsymbol{M}^n(\boldsymbol{x}_n^p)^\top\right]^\top$$

as required. Similarly

$$\boldsymbol{X}_s = (\boldsymbol{I}_m \otimes \boldsymbol{K}^s)\,\boldsymbol{Q}_{m,p_s} \tag{8.4.29}$$

by again using the partitioning of $\boldsymbol{Z}_\text{d}$ given in (8.4.22), noting that

$$\boldsymbol{Q}_{m,p_s}\text{Vec}\,(\boldsymbol{W}_s) = \text{Vec}\left(\boldsymbol{W}_s^\top\right)$$

via (B.5.4) and finally observing that

$$(\boldsymbol{I}_m \otimes \boldsymbol{K}^s)\,\text{Vec}\left(\boldsymbol{W}_s^\top\right) = \text{Vec}\left(\boldsymbol{K}^s\,\boldsymbol{W}_s^\top\right)$$

by (B.5.5) of Appendix B.5, as required.

The joint statistical model (8.4.26) is of the form

$$\boldsymbol{Y} = \boldsymbol{C}\boldsymbol{\beta} + \boldsymbol{\varepsilon}\,, \tag{8.4.30}$$

where $\boldsymbol{\beta}$ and $\boldsymbol{\varepsilon}$ are mutually independent with $\boldsymbol{\beta} \sim N(\boldsymbol{0},\boldsymbol{\Sigma}_\beta)$ and $\boldsymbol{\varepsilon} \sim N(\boldsymbol{0},\boldsymbol{\Sigma}_\varepsilon)$. Inference for this statistical model will require computation of its log likelihood function, which is the log of the probability density function of $\boldsymbol{Y}$. The vector $\boldsymbol{Y}$ is multivariate normally distributed assuming (8.4.30); therefore $\boldsymbol{Y}$ has log likelihood function (up to an additive constant)

$$\ell(\mathcal{P};\,\boldsymbol{Y}) = -\frac{1}{2}\,\ell n\,\det(\boldsymbol{\Sigma}_\varepsilon + \boldsymbol{C}\boldsymbol{\Sigma}_\beta\boldsymbol{C}^\top) - \frac{1}{2}\boldsymbol{Y}^\top(\boldsymbol{\Sigma}_\varepsilon + \boldsymbol{C}\boldsymbol{\Sigma}_\beta\boldsymbol{C}^\top)^{-1}\boldsymbol{Y} \tag{8.4.31}$$

where $\mathcal{P}$ denotes whatever parameters are used to define $\boldsymbol{\Sigma}_\beta$ and $\boldsymbol{\Sigma}_\varepsilon$.

If $\boldsymbol{Y}$ is high dimensional, the formula (8.4.31) for the log likelihood function is of limited use, unless the covariance matrix $\boldsymbol{\Sigma}_\varepsilon + \boldsymbol{C}\boldsymbol{\Sigma}_\beta\boldsymbol{C}^\top$ is structured in a way that facilitates lower-dimensional inverse and determinant calculations. Such dimension reduction does not apply to the joint statistical model (8.4.26). However, an alternative expression for the $\boldsymbol{Y}$ log likelihood function shows that dimension reduction is possible for this model. Consider an equivalent form for the inverse of the matrix $\boldsymbol{\Sigma}_\varepsilon + \boldsymbol{C}\boldsymbol{\Sigma}_\beta\boldsymbol{C}^\top$ given by Lemma B.4 of Appendix B:

$$\left(\boldsymbol{\Sigma}_\varepsilon + \boldsymbol{C}\boldsymbol{\Sigma}_\beta\boldsymbol{C}^\top\right)^{-1} = \boldsymbol{\Sigma}_\varepsilon^{-1} - \boldsymbol{\Sigma}_\varepsilon^{-1}\boldsymbol{C}\left(\boldsymbol{\Sigma}_\beta^{-1} + \boldsymbol{C}^\top\boldsymbol{\Sigma}_\varepsilon^{-1}\boldsymbol{C}\right)^{-1}\boldsymbol{C}^\top\boldsymbol{\Sigma}_\varepsilon^{-1} \tag{8.4.32}$$

assuming all inverses on the right-hand side exist. The desired dimension reduction uses (8.4.32) but further requires the matrix $\boldsymbol{C}^\top\boldsymbol{\Sigma}_\varepsilon^{-1}\boldsymbol{C}$ to be nonsingular, which holds if the coefficient matrix $\boldsymbol{C}$ is of full column rank. To avoid this potential issue, the matrix $\boldsymbol{C}^\top\boldsymbol{\Sigma}_\varepsilon^{-1}\boldsymbol{C}$ is replaced by

$$\boldsymbol{\Sigma}_\Gamma = \boldsymbol{C}^\top\boldsymbol{\Sigma}_\varepsilon^{-1}\boldsymbol{C} + \boldsymbol{\Gamma} \tag{8.4.33}$$

on the right-hand side of (8.4.32). Here $\boldsymbol{\Gamma} = diag(\gamma_1, \gamma_2, \ldots, \gamma_c)$ where c is the number of columns of $\boldsymbol{C}$ and $\gamma_i \geq 0$ for $i = 1, \ldots, c$. That is, if necessary, a nonnegative real number γ_i is added to the i^{th} diagonal element of $\boldsymbol{C}^\top \boldsymbol{\Sigma}_\varepsilon^{-1} \boldsymbol{C}$ for $i = 1, \ldots, c$. The chosen value of γ_i should be sufficiently large so that $\boldsymbol{\Sigma}_\Gamma$ is nonsingular, but otherwise as small as possible to minimize the impact of replacing the matrix $\boldsymbol{C}^\top \boldsymbol{\Sigma}_\varepsilon^{-1} \boldsymbol{C}$ by $\boldsymbol{\Sigma}_\Gamma$. The value 10^{-6} often performs well in practice. If $\boldsymbol{C}$ is of full column rank, $\gamma_i = 0$ should be chosen for $i = 1, \ldots, c$ unless the matrix $\boldsymbol{C}^\top \boldsymbol{\Sigma}_\varepsilon^{-1} \boldsymbol{C}$ is ill-conditioned.

Using $\boldsymbol{\Sigma}_\Gamma$, (B.5.2) of Appendix B is invoked to obtain

$$\left(\boldsymbol{\Sigma}_\beta^{-1} + \boldsymbol{C}^\top \boldsymbol{\Sigma}_\varepsilon^{-1} \boldsymbol{C}\right)^{-1} \approx \left(\boldsymbol{\Sigma}_\beta^{-1} + \boldsymbol{\Sigma}_\Gamma\right)^{-1} = \boldsymbol{\Sigma}_\Gamma^{-1} - \boldsymbol{\Sigma}_\Gamma^{-1} \left(\boldsymbol{\Sigma}_\beta + \boldsymbol{\Sigma}_\Gamma^{-1}\right)^{-1} \boldsymbol{\Sigma}_\Gamma^{-1}. \tag{8.4.34}$$

Substituting (8.4.34) into (8.4.32) results in the $\boldsymbol{\Sigma}_\Gamma$-based approximate log likelihood function:

$$\begin{aligned} \ell_{\boldsymbol{\Gamma}}(\mathcal{P}; \boldsymbol{Y}) = &- \frac{1}{2}\, \ell n \det(\boldsymbol{\Sigma}_\varepsilon) - \frac{1}{2}\, \ell n \det(\boldsymbol{\Sigma}_\Gamma) \\ &- \frac{1}{2}\left[(\boldsymbol{Y} - \boldsymbol{C}\widehat{\boldsymbol{\beta}})^\top \boldsymbol{\Sigma}_\varepsilon^{-1} (\boldsymbol{Y} - \boldsymbol{C}\widehat{\boldsymbol{\beta}}) + \widehat{\boldsymbol{\beta}}^\top \boldsymbol{\Gamma} \widehat{\boldsymbol{\beta}}\right] \\ &- \frac{1}{2}\, \ell n \det(\boldsymbol{\Sigma}_\beta + \boldsymbol{\Sigma}_\Gamma^{-1}) - \frac{1}{2} \widehat{\boldsymbol{\beta}}^\top (\boldsymbol{\Sigma}_\beta + \boldsymbol{\Sigma}_\Gamma^{-1})^{-1} \widehat{\boldsymbol{\beta}} \end{aligned} \tag{8.4.35}$$

where

$$\widehat{\boldsymbol{\beta}} = \boldsymbol{\Sigma}_\Gamma^{-1} \boldsymbol{C}^\top \boldsymbol{\Sigma}_\varepsilon^{-1} \boldsymbol{Y} \tag{8.4.36}$$

is the (approximate) generalized least squares estimator of $\boldsymbol{\beta}$. Note that when $\boldsymbol{C}$ is of full column rank and $\boldsymbol{\Gamma} = \mathbf{0}_{c,c}$ is a feasible choice, algebra gives $\ell(\mathcal{P}; \boldsymbol{Y}) = \ell_{\mathbf{0}_{c,c}}(\mathcal{P}; \boldsymbol{Y})$.

In this formulation, $\widehat{\boldsymbol{\beta}}$ assumes the role of "data" in the log likelihood calculation if it does not depend on unknown parameters. Using (8.4.36), $\widehat{\boldsymbol{\beta}} \sim N_c(\mathbf{0}_c, \boldsymbol{\Sigma}_{\widehat{\beta}})$, where

$$\boldsymbol{\Sigma}_{\widehat{\beta}} = (\boldsymbol{I}_c - \boldsymbol{\Sigma}_\Gamma^{-1} \boldsymbol{\Gamma}) \boldsymbol{\Sigma}_\beta (\boldsymbol{I}_c - \boldsymbol{\Gamma} \boldsymbol{\Sigma}_\Gamma^{-1}) + \boldsymbol{\Sigma}_\Gamma^{-1} - \boldsymbol{\Sigma}_\Gamma^{-1} \boldsymbol{\Gamma} \boldsymbol{\Sigma}_\Gamma^{-1}.$$

If the matrix $\boldsymbol{C}$ is of full column rank, allowing the choice $\boldsymbol{\Gamma} = \mathbf{0}_{c,c}$, then

$$\boldsymbol{\Sigma}_{\widehat{\beta}} = \boldsymbol{\Sigma}_\beta + \boldsymbol{\Sigma}_\Gamma^{-1}$$

and the last line of (8.4.35) is the log likelihood of the "data" $\widehat{\boldsymbol{\beta}}$ up to an additive constant. This holds approximately when $\boldsymbol{\Gamma}$ is "small," as is expected in essentially every application. Substantial dimension reduction in the calculation of (8.4.35) is therefore possible when $\boldsymbol{\beta}$ is of considerably lower dimension than $\boldsymbol{Y}$, and $\boldsymbol{\Sigma}_\varepsilon$ is fixed.

A second circumstance where dimension reduction is possible is when $\boldsymbol{\Sigma}_\varepsilon$ has a "simple" parameterization. Suppose that $\boldsymbol{\Sigma}_\varepsilon = \lambda^{-1} \boldsymbol{\Sigma}_\varepsilon^0$ for a given $\boldsymbol{\Sigma}_\varepsilon^0$; applying (8.4.33) with $\boldsymbol{\Gamma} = \lambda r_0 \boldsymbol{I}_c$ to calculate $\widehat{\boldsymbol{\beta}}$ in (8.4.36) gives

$$\widehat{\boldsymbol{\beta}} = \left(\boldsymbol{C}^\top \left(\boldsymbol{\Sigma}_\varepsilon^0\right)^{-1} \boldsymbol{C} + r_0 \boldsymbol{I}_c\right)^{-1} \boldsymbol{C}^\top \left(\boldsymbol{\Sigma}_\varepsilon^0\right)^{-1} \boldsymbol{Y}$$

which does not depend on λ. The second line of (8.4.35) is given by

$$\lambda\left((\boldsymbol{Y}-\boldsymbol{C}\widehat{\boldsymbol{\beta}})^\top\left(\boldsymbol{\Sigma}_\varepsilon^0\right)^{-1}(\boldsymbol{Y}-\boldsymbol{C}\widehat{\boldsymbol{\beta}})+r_0\widehat{\boldsymbol{\beta}}^\top\widehat{\boldsymbol{\beta}}\right)$$

and only a single computation of the fixed quadratic forms is required. The parameterization of the joint statistical model (8.4.26) also permits the desired dimension reduction, as demonstrated below.

Noting that $\boldsymbol{Q}_{s,r}=\boldsymbol{Q}_{r,s}^\top$ for vec-permutation matrices, calculation gives

$$\boldsymbol{C}^\top\boldsymbol{\Sigma}_\varepsilon^{-1}\boldsymbol{C}=$$
$$\begin{pmatrix}\lambda_{p,\varepsilon}\boldsymbol{Q}_{p_\delta+p_s,n}\boldsymbol{F}_{DK}^\top\boldsymbol{P}^{p,n}\boldsymbol{F}_{DK}\boldsymbol{Q}_{n,p_\delta+p_s} & \boldsymbol{0}_{n(p_\delta+p_s),mp_s}\\ \boldsymbol{0}_{mp_s,n(p_\delta+p_s)} & \lambda_{s,\varepsilon}\boldsymbol{Q}_{p_s,m}\left(\boldsymbol{I}_m\otimes(\boldsymbol{K}^s)^\top\boldsymbol{K}^s\right)\boldsymbol{Q}_{m,p_s}\end{pmatrix}.$$

In this scenario, the coefficient matrix $\boldsymbol{X}_s$ is of full column rank as long as $p_s\le\min\{m_s,m-1\}$, because then the columns of $\boldsymbol{K}^s$ are orthogonal by construction. However, the coefficient matrix $\boldsymbol{X}_{p,\mathrm{d}}$ is not guaranteed to be of full column rank even with the requirement $p_\delta\le\min_{i=1,\ldots,n}\left(m_{p,i}\right)-p_s$. Its column rank depends on the discrepancy basis vectors constructed and their relationship to the simulator basis vectors. Specifically, it is possible that at least one of the matrices $\left[\boldsymbol{D}_i^p\ \boldsymbol{K}_i^s\right]$ for $i\in\{1,\ldots,n\}$ is not of full column rank, in which case the matrix $\boldsymbol{F}_{DK}$ will be of reduced column rank. This suggests that only the upper-left block diagonal element of $\boldsymbol{C}^\top\boldsymbol{\Sigma}_\varepsilon^{-1}\boldsymbol{C}$ needs to be adjusted to form $\boldsymbol{\Sigma}_\Gamma$. Taking

$$\gamma_1=\gamma_2=\cdots=\gamma_{n(p_\delta+p_s)}=\lambda_{p,\varepsilon}r_0\,,\text{ and }\gamma_{n(p_\delta+p_s)+1}=\cdots=\gamma_{np_\delta+(n+m)p_s}=0 \tag{8.4.37}$$

as the diagonal elements of Γ in (8.4.33) and noting that $\boldsymbol{Q}_{s,r}\boldsymbol{Q}_{r,s}=\boldsymbol{I}_{rs}$ for vec-permutation matrices, this adjustment involves replacing $\boldsymbol{F}_{DK}^\top\boldsymbol{P}^{p,n}\boldsymbol{F}_{DK}$ with

$$\begin{aligned}\boldsymbol{\Lambda}_{DK}&\equiv\boldsymbol{F}_{DK}^\top\boldsymbol{P}^{p,n}\boldsymbol{F}_{DK}+r_0\boldsymbol{I}_{n(p_\delta+p_s)}\\&=diag\left(\begin{pmatrix}\left(\boldsymbol{D}_i^p\right)^\top\boldsymbol{P}_i^{p,n}\boldsymbol{D}_i^p+r_0\boldsymbol{I}_{p_\delta} & \left(\boldsymbol{D}_i^p\right)^\top\boldsymbol{P}_i^{p,n}\boldsymbol{K}_i^s\\ \left(\boldsymbol{K}_i^s\right)^\top\boldsymbol{P}_i^{p,n}\boldsymbol{D}_i^p & \left(\boldsymbol{K}_i^s\right)^\top\boldsymbol{P}_i^{p,n}\boldsymbol{K}_i^s+r_0\boldsymbol{I}_{p_s}\end{pmatrix},\ i=1,\ldots,n\right).\end{aligned} \tag{8.4.38}$$

Using $\boldsymbol{\Lambda}_{DK}$ and the fact that $(\boldsymbol{K}^s)^\top\boldsymbol{K}^s=m^{-1}\left(\widetilde{\boldsymbol{\Sigma}}_{11}^s\right)^2$, the matrix $\boldsymbol{\Sigma}_{\Gamma,\mathrm{d}}$ is obtained:

$$\boldsymbol{\Sigma}_{\Gamma,\mathrm{d}}=\begin{pmatrix}\lambda_{p,\varepsilon}\boldsymbol{Q}_{p_\delta+p_s,n}\boldsymbol{\Lambda}_{DK}\boldsymbol{Q}_{n,p_\delta+p_s} & \boldsymbol{0}_{n(p_\delta+p_s),mp_s}\\ \boldsymbol{0}_{mp_s,n(p_\delta+p_s)} & \lambda_{s,\varepsilon}\,m^{-1}\left(\left(\widetilde{\boldsymbol{\Sigma}}_{11}^s\right)^2\otimes\boldsymbol{I}_m\right)\end{pmatrix}. \tag{8.4.39}$$

The lower-right block diagonal element of $\boldsymbol{C}^\top\boldsymbol{\Sigma}_\varepsilon^{-1}\boldsymbol{C}$ was reduced further in this expression using (B.5.6) of Appendix B.5. It follows that

$$\boldsymbol{\Sigma}_{\Gamma,\mathrm{d}}^{-1} = \begin{pmatrix} \lambda_{p,\varepsilon}^{-1}\boldsymbol{Q}_{p_\delta+p_s,n}\boldsymbol{\Lambda}_{DK}^{-1}\boldsymbol{Q}_{n,p_\delta+p_s} & \boldsymbol{0}_{n(p_\delta+p_s),mp_s} \\ \boldsymbol{0}_{mp_s,n(p_\delta+p_s)} & \lambda_{s,\varepsilon}^{-1}\, m\left(\left(\widetilde{\boldsymbol{\Sigma}}_{11}^{s}\right)^{-2} \otimes \boldsymbol{I}_m\right) \end{pmatrix}. \tag{8.4.40}$$

From (8.4.38), it is evident that computing $\boldsymbol{\Sigma}_{\Gamma,\mathrm{d}}^{-1}$ involves inverting n square matrices of dimension $(p_\delta + p_s)$ and the square matrix $\widetilde{\boldsymbol{\Sigma}}_{11}^{s}$ of dimension p_s. These matrices are all fixed, allowing their inverses to be computed once, stored, and accessed as needed.

In the application of (8.4.26), $\widehat{\boldsymbol{\beta}}$ in (8.4.36) will be designated $\widehat{\boldsymbol{Z}}_{\mathrm{d}}$ and

$$\boldsymbol{C}^\top \boldsymbol{\Sigma}_\varepsilon^{-1}\boldsymbol{Y} = \begin{pmatrix} \lambda_{p,\varepsilon}\,\boldsymbol{Q}_{p_\delta+p_s,n}\boldsymbol{F}_{DK}^\top \boldsymbol{P}^{p,n}\boldsymbol{Y}_{p,n} \\ \lambda_{s,\varepsilon}\left((\boldsymbol{K}^s)^\top \otimes \boldsymbol{I}_m\right)\boldsymbol{Q}_{m_s,m}\boldsymbol{Y}_{s,n,\varepsilon} \end{pmatrix},$$

where (B.5.6) was invoked to obtain the bottom entry. The vector $\widehat{\boldsymbol{Z}}_{\mathrm{d}}$ follows:

$$\widehat{\boldsymbol{Z}}_{\mathrm{d}} = \begin{pmatrix} \boldsymbol{Q}_{p_\delta+p_s,n}\boldsymbol{\Lambda}_{DK}^{-1}\boldsymbol{F}_{DK}^\top \boldsymbol{P}^{p,n}\boldsymbol{Y}_{p,n} \\ m\,\boldsymbol{Q}_{p_s,m}\left(\boldsymbol{I}_m \otimes \left(\widetilde{\boldsymbol{\Sigma}}_{11}^{s}\right)^{-2}(\boldsymbol{K}^s)^\top\right)\boldsymbol{Y}_{s,n,\varepsilon} \end{pmatrix}, \tag{8.4.41}$$

where again (B.5.6) was invoked to obtain the bottom entry.

Both entries of $\widehat{\boldsymbol{Z}}_{\mathrm{d}}$ are subject to further simplification. Define the $p_\delta \times 1$ vectors $\{\widetilde{\boldsymbol{V}}^p(\boldsymbol{x}_i^p)\}$ and the $p_s \times 1$ vectors $\{\widetilde{\boldsymbol{U}}_{\mathrm{d}}^s(\boldsymbol{x}_i^p)\}$ for $i = 1, \ldots, n$:

$$\begin{pmatrix} \widetilde{\boldsymbol{V}}^p(\boldsymbol{x}_i^p) \\ \widetilde{\boldsymbol{U}}_{\mathrm{d}}^s(\boldsymbol{x}_i^p) \end{pmatrix} =$$

$$\begin{pmatrix} \left(\boldsymbol{D}_i^p\right)^\top \boldsymbol{P}_i^{p,n}\boldsymbol{D}_i^p + r_0\boldsymbol{I}_{p_\delta} & \left(\boldsymbol{D}_i^p\right)^\top \boldsymbol{P}_i^{p,n}\boldsymbol{K}_i^s \\ \left(\boldsymbol{K}_i^s\right)^\top \boldsymbol{P}_i^{p,n}\boldsymbol{D}_i^p & \left(\boldsymbol{K}_i^s\right)^\top \boldsymbol{P}_i^{p,n}\boldsymbol{K}_i^s + r_0\boldsymbol{I}_{p_s} \end{pmatrix}^{-1} \begin{pmatrix} \left(\boldsymbol{D}_i^p\right)^\top \\ \left(\boldsymbol{K}_i^s\right)^\top \end{pmatrix} \boldsymbol{P}_i^{p,n}\boldsymbol{Y}^{p,n}(\boldsymbol{x}_i^p).$$

Collecting these vectors into the $n \times p_\delta$ matrix $\widetilde{\boldsymbol{V}}_p$ and the $n \times p_s$ matrix $\widetilde{\boldsymbol{U}}_{\mathrm{d},s}$

$$\widetilde{\boldsymbol{V}}_p = \left[\widetilde{\boldsymbol{V}}^p(\boldsymbol{x}_1^p)\ \widetilde{\boldsymbol{V}}^p(\boldsymbol{x}_2^p) \cdots \widetilde{\boldsymbol{V}}^p(\boldsymbol{x}_n^p)\right]^\top, \tag{8.4.42}$$

$$\widetilde{\boldsymbol{U}}_{\mathrm{d},s} = \left[\widetilde{\boldsymbol{U}}_{\mathrm{d}}^s(\boldsymbol{x}_1^p)\ \widetilde{\boldsymbol{U}}_{\mathrm{d}}^s(\boldsymbol{x}_2^p) \cdots \widetilde{\boldsymbol{U}}_{\mathrm{d}}^s(\boldsymbol{x}_n^p)\right]^\top, \tag{8.4.43}$$

gives

$$\boldsymbol{\Lambda}_{DK}^{-1}\boldsymbol{F}_{DK}^\top \boldsymbol{P}^{p,n}\boldsymbol{Y}_{p,n} = \mathrm{Vec}\left(\left[\widetilde{\boldsymbol{V}}_p\ \widetilde{\boldsymbol{U}}_{\mathrm{d},s}\right]^\top\right). \tag{8.4.44}$$

Property (B.5.4) can be invoked to simplify the top entry of $\widehat{\boldsymbol{Z}}_{\mathrm{d}}$. Utilizing (8.4.44) and (B.5.4)

$$\boldsymbol{Q}_{p_\delta+p_s,n}\boldsymbol{\Lambda}_{DK}^{-1}\boldsymbol{F}_{DK}^\top \boldsymbol{P}^{p,n}\boldsymbol{Y}_{p,n} = \mathrm{Vec}\left(\left[\widetilde{\boldsymbol{V}}_p\ \widetilde{\boldsymbol{U}}_{\mathrm{d},s}\right]\right). \tag{8.4.45}$$

Define the $p_s \times 1$ vectors $\{\widetilde{\boldsymbol{W}}^s(\boldsymbol{x}_i^s, \boldsymbol{t}_i)\}$ for $i = 1, \ldots, m$:

$$\widetilde{\boldsymbol{W}}^s(\boldsymbol{x}_i^s, \boldsymbol{t}_i) = m\left(\widetilde{\boldsymbol{\Sigma}}_{11}^s\right)^{-2} (\boldsymbol{K}^s)^\top \boldsymbol{Y}^{s,n,\varepsilon}(\boldsymbol{x}_i^s, \boldsymbol{t}_i) .$$

Collecting these vectors into the $m \times p_s$ matrix $\widetilde{\boldsymbol{W}}_s$

$$\widetilde{\boldsymbol{W}}_s = \left[\widetilde{\boldsymbol{W}}^s(\boldsymbol{x}_1^s, \boldsymbol{t}_1)\ \widetilde{\boldsymbol{W}}^s(\boldsymbol{x}_2^s, \boldsymbol{t}_2) \cdots \widetilde{\boldsymbol{W}}^s(\boldsymbol{x}_m^s, \boldsymbol{t}_m)\right]^\top , \tag{8.4.46}$$

gives

$$m\left(\boldsymbol{I}_m \otimes \left(\widetilde{\boldsymbol{\Sigma}}_{11}^s\right)^{-2} (\boldsymbol{K}^s)^\top\right) \boldsymbol{Y}_{s,n,\varepsilon} = \text{Vec}\left(\widetilde{\boldsymbol{W}}_s^\top\right)$$

by using (B.5.5). Invoking (B.5.4) yields

$$m\,\boldsymbol{Q}_{p_s,m}\left(\boldsymbol{I}_m \otimes \left(\widetilde{\boldsymbol{\Sigma}}_{11}^s\right)^{-2} (\boldsymbol{K}^s)^\top\right) \boldsymbol{Y}_{s,n,\varepsilon} = \text{Vec}\left(\widetilde{\boldsymbol{W}}_s\right) . \tag{8.4.47}$$

Substituting (8.4.45) and (8.4.47) into the top and bottom entries of (8.4.41), respectively, gives

$$\widehat{\boldsymbol{Z}}_\text{d} = \begin{pmatrix} \text{Vec}\left(\left[\widetilde{\boldsymbol{V}}_p\ \widetilde{\boldsymbol{U}}_{\text{d},s}\right]\right) \\ \text{Vec}\left(\widetilde{\boldsymbol{W}}_s\right) \end{pmatrix} . \tag{8.4.48}$$

The $(n(p_\delta + p_s) + mp_s) \times 1$ vector $\widehat{\boldsymbol{Z}}_\text{d}$ does not depend on any unknown parameters. If the simulator and discrepancy basis representations are relatively efficient, namely, $p_s \ll m_s$ and $(p_\delta + p_s) \ll (m_y/n)$, then $\widehat{\boldsymbol{Z}}_\text{d}$ will have substantially reduced dimension compared with the physical experiment data and calculated simulation output used to compute it. The development in Sect. 8.5.1 will clarify the added efficiency obtained by using reduced "data" $\widehat{\boldsymbol{Z}}_\text{d}$ in log likelihood calculations.

Combining and simplifying (8.4.26), (8.4.27), (8.4.28), (8.4.29), (8.4.42), (8.4.43), (8.4.46), and (8.4.48), along with (B.5.4) for the experimental data block and (B.5.5) for the simulator block, yield

$$(\boldsymbol{Y} - \boldsymbol{C}\widehat{\boldsymbol{\beta}})^\top \boldsymbol{\Sigma}_\varepsilon^{-1} (\boldsymbol{Y} - \boldsymbol{C}\widehat{\boldsymbol{\beta}}) = \lambda_{p,\varepsilon} \sum_{i=1}^{n} \left\| \boldsymbol{Y}^{p,n}(\boldsymbol{x}_i^p) - \boldsymbol{D}_i^p \widetilde{\boldsymbol{V}}^p(\boldsymbol{x}_i^p) - \boldsymbol{K}_i^s \widetilde{\boldsymbol{U}}_\text{d}^s(\boldsymbol{x}_i^p) \right\|_{\boldsymbol{P}_i^{p,n}}^2$$
$$+ \lambda_{s,\varepsilon} \sum_{i=1}^{m} \left\| \boldsymbol{Y}^{s,n,\epsilon}(\boldsymbol{x}_i^s, \boldsymbol{t}_i) - \boldsymbol{K}^s \widetilde{\boldsymbol{W}}^s(\boldsymbol{x}_i^s, \boldsymbol{t}_i) \right\|_{\boldsymbol{I}_{m_s}}^2 \tag{8.4.49}$$

where $||\boldsymbol{x}||_{\boldsymbol{A}}^2 = \boldsymbol{x}^\top \boldsymbol{A} \boldsymbol{x}$ is the quadratic form operator. Employing (8.4.37),

$$\boldsymbol{\Gamma} = diag(\lambda_{p,\varepsilon} r_0 \boldsymbol{I}_{n(p_\delta + p_s)}, \boldsymbol{0}_{mp_s, mp_s})$$

and utilizing (8.4.48) produce

$$\widehat{\boldsymbol{\beta}}^\top \boldsymbol{\Gamma} \widehat{\boldsymbol{\beta}} = \lambda_{p,\varepsilon}\, r_0 \sum_{i=1}^{n} \left(\left\| \widetilde{\boldsymbol{V}}^p(\boldsymbol{x}_i^p) \right\|_{\boldsymbol{I}_{p_\delta}}^2 + \left\| \widetilde{\boldsymbol{U}}_\text{d}^s(\boldsymbol{x}_i^p) \right\|_{\boldsymbol{I}_{p_s}}^2 \right) . \tag{8.4.50}$$

Higdon et al. (2008) neglected this term under the presumption that r_0 is small relative to the sum of the (estimated) squared coefficient norms. The sums in (8.4.49) and (8.4.50) do not depend on any unknown parameters, implying they can be computed once, stored, and accessed as needed.

Plugging (8.4.27), (8.4.39), (8.4.48), (8.4.49), (8.4.50), (8.4.23), and (8.4.40) into (8.4.35), the (approximate) log likelihood function of $\boldsymbol{\Omega}^{\text{d, all}}$ is given by

$$\begin{aligned}
&\ell_{\text{d}}(\boldsymbol{\Omega}^{\text{d, all}}; \boldsymbol{Y}_{p,n}, \boldsymbol{Y}_{s,n,\varepsilon}, r_0) = \\
&\qquad \frac{m_y - n(p_\delta + p_s)}{2}\, \ell n\left(\lambda_{p,\varepsilon}\right) + \frac{m(m_s - p_s)}{2}\, \ell n\left(\lambda_{s,\varepsilon}\right) \\
&\qquad - \lambda_{p,\varepsilon}\left[\frac{1}{2}\sum_{i=1}^{n}\left\|\boldsymbol{Y}^{p,n}(\boldsymbol{x}_i^p) - \boldsymbol{D}_i^p\widetilde{\boldsymbol{V}}^p(\boldsymbol{x}_i^p) - \boldsymbol{K}_i^s\widetilde{\boldsymbol{U}}_{\text{d}}^s(\boldsymbol{x}_i^p)\right\|^2_{\boldsymbol{P}_i^{p,n}}\right. \\
&\qquad \left. + \frac{r_0}{2}\sum_{i=1}^{n}\left(\left\|\widetilde{\boldsymbol{V}}^p(\boldsymbol{x}_i^p)\right\|^2_{\boldsymbol{I}_{p_\delta}} + \left\|\widetilde{\boldsymbol{U}}_{\text{d}}^s(\boldsymbol{x}_i^p)\right\|^2_{\boldsymbol{I}_{p_s}}\right)\right] \\
&\qquad - \lambda_{s,\varepsilon}\left[\frac{1}{2}\sum_{i=1}^{m}\left\|\boldsymbol{Y}^{s,n,\epsilon}(\boldsymbol{x}_i^s, \boldsymbol{t}_i) - \boldsymbol{K}^s\widetilde{\boldsymbol{W}}^s(\boldsymbol{x}_i^s, \boldsymbol{t}_i)\right\|^2_{\boldsymbol{I}_{m_s}}\right] \\
&\qquad - \frac{1}{2}\,\ell n \det\left(\boldsymbol{\Sigma}_{Z,\text{d}} + \boldsymbol{\Sigma}_{\Gamma,\text{d}}^{-1}\right) - \frac{1}{2}\widehat{\boldsymbol{Z}}_{\text{d}}^{\top}\left(\boldsymbol{\Sigma}_{Z,\text{d}} + \boldsymbol{\Sigma}_{\Gamma,\text{d}}^{-1}\right)^{-1}\widehat{\boldsymbol{Z}}_{\text{d}}\,. \qquad (8.4.51)
\end{aligned}$$

Here $\boldsymbol{\Omega}^{\text{d, all}} = \{\boldsymbol{\Omega}^{p,\text{d},\varepsilon}, \lambda_{s,\varepsilon}\}$ is the collection of all unknown parameters appearing in the joint statistical model (8.4.26), and r_0 is defined as in (8.4.37). Additive constants not depending on $\boldsymbol{\Omega}^{\text{d, all}}$ have been excluded from (8.4.51) as they have no effect on statistical inference.

8.4.3.2 Joint Statistical Model Assuming No Simulator Discrepancy

When the simulator discrepancy is *assumed to be identically zero*, i.e., (8.4.20) holds, the joint statistical model will be specified in terms of the vector

$$\boldsymbol{Z}_{\text{nod}} = \begin{pmatrix} \text{Vec}\,(\boldsymbol{U}_s) \\ \text{Vec}\,(\boldsymbol{W}_s) \end{pmatrix} \qquad (8.4.52)$$

of length $(n+m)p_s$ which omits the discrepancy coefficients. Because $\boldsymbol{Z}_{\text{nod}}$ is constructed from mean-zero GPs, conditionally on $\boldsymbol{\Omega}^{p,\text{nod}}$, it is multivariate normally distributed with zero mean vector and covariance matrix

$$\boldsymbol{\Sigma}_{Z,\text{nod}} = \begin{pmatrix} \boldsymbol{\Sigma}_U & \boldsymbol{\Sigma}_{UW} \\ \boldsymbol{\Sigma}_{UW}^{\top} & \boldsymbol{\Sigma}_W \end{pmatrix} \qquad (8.4.53)$$

or, in an earlier notation,

$$\left[\boldsymbol{Z}_{\text{nod}} \,\middle|\, \boldsymbol{\Omega}^{p,\text{nod}}\right] \sim N_{(n+m)p_s}\left(\boldsymbol{0}_{(n+m)p_s}, \boldsymbol{\Sigma}_{Z,\text{nod}}\right) .$$

The joint statistical model takes the following form:

$$\begin{pmatrix} \boldsymbol{Y}_{p,n} \\ \boldsymbol{Y}_{s,n,\varepsilon} \end{pmatrix} = \begin{pmatrix} \boldsymbol{X}_{p,\text{nod}} & \boldsymbol{0}_{m_y,mp_s} \\ \boldsymbol{0}_{m_s m,np_s} & \boldsymbol{X}_s \end{pmatrix} \boldsymbol{Z}_{\text{nod}} + \begin{pmatrix} \boldsymbol{E}_{p,n} \\ \boldsymbol{E}_s \end{pmatrix} . \tag{8.4.54}$$

The vectors $\boldsymbol{Z}_{\text{nod}}$ and $\left(\boldsymbol{E}_{p,n}^\top, \boldsymbol{E}_s^\top\right)^\top$ are assumed to be mutually independent.

Collecting the regression matrices of (8.4.20) into a $m_y \times np_s$ block diagonal matrix,

$$\boldsymbol{F}_K = diag(\boldsymbol{K}_1^s, \boldsymbol{K}_2^s, \ldots, \boldsymbol{K}_n^s) ,$$

$\boldsymbol{X}_{p,\text{nod}}$ is given by

$$\boldsymbol{X}_{p,\text{nod}} = \boldsymbol{F}_K \, \boldsymbol{Q}_{n,p_s} . \tag{8.4.55}$$

This is seen by using the partitioning of $\boldsymbol{Z}_{\text{nod}}$ given in (8.4.52), noting that

$$\boldsymbol{Q}_{n,p_s} \text{Vec}\,(\boldsymbol{U}_s) = \text{Vec}\left(\boldsymbol{U}_s^\top\right)$$

via (B.5.4) and finally observing that

$$\boldsymbol{F}_K \text{Vec}\left(\boldsymbol{U}_s^\top\right) = \left[\boldsymbol{M}_0^n(\boldsymbol{x}_1^p)^\top \ \boldsymbol{M}_0^n(\boldsymbol{x}_2^p)^\top \ \cdots \ \boldsymbol{M}_0^n(\boldsymbol{x}_n^p)^\top\right]^\top$$

as required.

The joint statistical model (8.4.54) has the form (8.4.30), and thus the construction of the log likelihood function parallels the previous results for joint statistical model (8.4.26). Only the matrix blocks affected by the statistical model of the physical observations change, as seen by comparing (8.4.54) and (8.4.26). Matrix blocks determined by the statistical model of the simulation output remain the same as above and are reproduced in simplied form below. The modified $\boldsymbol{C}^\top \boldsymbol{\Sigma}_\varepsilon^{-1} \boldsymbol{C}$ matrix is given by

$$\boldsymbol{C}^\top \boldsymbol{\Sigma}_\varepsilon^{-1} \boldsymbol{C} = \begin{pmatrix} \lambda_{p,\varepsilon} \boldsymbol{Q}_{p_s,n} \boldsymbol{F}_K^\top \boldsymbol{P}^{p,n} \boldsymbol{F}_K \boldsymbol{Q}_{n,p_s} & \boldsymbol{0}_{np_s,mp_s} \\ \boldsymbol{0}_{mp_s,np_s} & \lambda_{s,\varepsilon}\, m^{-1} \left(\left(\widetilde{\boldsymbol{\Sigma}}_{11}^s\right)^2 \otimes \boldsymbol{I}_m\right) \end{pmatrix} .$$

The coefficient matrix $\boldsymbol{X}_{p,\text{nod}}$ will generally be of full column rank because the matrices $\boldsymbol{K}_i^s$ for $i = 1, \ldots, n$ should be of full column rank as long as the value of p_s chosen satisfies $p_s \leq \min_{i=1,\ldots,n} m_{p,i}$. However, this inequality could be violated in the rare circumstance of a sparse set of physical observations paired with a more complex simulator. Furthermore, because the columns of $\boldsymbol{K}_i^s$ are interpolations of the simulator basis vectors onto the index variable(s) settings corresponding to the i^{th} physical experiment, the rare possibility of at least one of these coefficient matrices having reduced column rank (thus rendering $\boldsymbol{F}_K$ of reduced column rank) is allowed for in the formulation of $\boldsymbol{\Sigma}_{\Gamma,\text{nod}}$. Analogous to the formulation of $\boldsymbol{\Sigma}_{\Gamma,\text{d}}$, taking

$$\gamma_1 = \gamma_2 = \cdots = \gamma_{np_s} = \lambda_{p,\varepsilon} r_0 \, , \ \text{and} \ \gamma_{np_s+1} = \cdots = \gamma_{(n+m)p_s} = 0 \tag{8.4.56}$$

the adjustment to the upper-left block diagonal element of $\boldsymbol{C}^\top \boldsymbol{\Sigma}_\varepsilon^{-1} \boldsymbol{C}$ involves replacing $\boldsymbol{F}_K^\top \boldsymbol{P}^{p,n} \boldsymbol{F}_K$ with $\boldsymbol{\Lambda}_K$:

$$\boldsymbol{\Lambda}_K = diag\left(\left(\boldsymbol{K}_i^s\right)^\top \boldsymbol{P}_i^{p,n} \boldsymbol{K}_i^s + r_0 \boldsymbol{I}_{p_s}\,,\ i = 1, \ldots, n\right). \tag{8.4.57}$$

As discussed above, typical applications in this setting will allow $r_0 = 0$ to be chosen.

Using $\boldsymbol{\Lambda}_K$, the matrices $\boldsymbol{\Sigma}_{\Gamma,\text{nod}}$ and $\boldsymbol{\Sigma}_{\Gamma,\text{nod}}^{-1}$ are

$$\boldsymbol{\Sigma}_{\Gamma,\text{nod}} = \begin{pmatrix} \lambda_{p,\varepsilon} \boldsymbol{Q}_{p_s,n} \boldsymbol{\Lambda}_K \boldsymbol{Q}_{n,p_s} & \boldsymbol{0}_{np_s,mp_s} \\ \boldsymbol{0}_{mp_s,np_s} & \lambda_{s,\varepsilon}\, m^{-1} \left(\left(\widetilde{\boldsymbol{\Sigma}}_{11}^s\right)^2 \otimes \boldsymbol{I}_m\right) \end{pmatrix} \tag{8.4.58}$$

$$\boldsymbol{\Sigma}_{\Gamma,\text{nod}}^{-1} = \begin{pmatrix} \lambda_{p,\varepsilon}^{-1} \boldsymbol{Q}_{p_s,n} \boldsymbol{\Lambda}_K^{-1} \boldsymbol{Q}_{n,p_s} & \boldsymbol{0}_{np_s,mp_s} \\ \boldsymbol{0}_{mp_s,np_s} & \lambda_{s,\varepsilon}^{-1}\, m \left(\left(\widetilde{\boldsymbol{\Sigma}}_{11}^s\right)^{-2} \otimes \boldsymbol{I}_m\right) \end{pmatrix}. \tag{8.4.59}$$

From (8.4.57), it is evident that computing $\boldsymbol{\Sigma}_{\Gamma,\text{nod}}^{-1}$ involves inverting n square matrices of dimension p_s and the square matrix $\widetilde{\boldsymbol{\Sigma}}_{11}^s$ of dimension p_s. These matrices are all fixed, allowing their inverses to be computed once, stored, and accessed as needed.

In the application of (8.4.54), $\widehat{\boldsymbol{\beta}}$ in (8.4.36) will be designated $\widehat{\boldsymbol{Z}}_\text{nod}$ and

$$\boldsymbol{C}^\top \boldsymbol{\Sigma}_\varepsilon^{-1} \boldsymbol{Y} = \begin{pmatrix} \lambda_{p,\varepsilon}\, \boldsymbol{Q}_{p_s,n} \boldsymbol{F}_K^\top \boldsymbol{P}^{p,n} \boldsymbol{Y}_{p,n} \\ \lambda_{s,\varepsilon} \left((\boldsymbol{K}^s)^\top \otimes \boldsymbol{I}_m\right) \boldsymbol{Q}_{m_s,m} \boldsymbol{Y}_{s,n,\varepsilon} \end{pmatrix}.$$

The vector $\widehat{\boldsymbol{Z}}_\text{nod}$ with bottom entry obtained from (B.5.6) and the substitution (8.4.47) follows:

$$\widehat{\boldsymbol{Z}}_\text{nod} = \begin{pmatrix} \boldsymbol{Q}_{p_s,n} \boldsymbol{\Lambda}_K^{-1} \boldsymbol{F}_K^\top \boldsymbol{P}^{p,n} \boldsymbol{Y}_{p,n} \\ \text{Vec}\left(\widetilde{\boldsymbol{W}}_s\right) \end{pmatrix}. \tag{8.4.60}$$

Define the $p_s \times 1$ vectors $\{\widetilde{\boldsymbol{U}}_\text{nod}^s(\boldsymbol{x}_i^p)\}$ for $i = 1, \ldots, n$:

$$\widetilde{\boldsymbol{U}}_\text{nod}^s(\boldsymbol{x}_i^p) = \left(\left(\boldsymbol{K}_i^s\right)^\top \boldsymbol{P}_i^{p,n} \boldsymbol{K}_i^s + r_0 \boldsymbol{I}_{p_s}\right)^{-1} \left(\boldsymbol{K}_i^s\right)^\top \boldsymbol{P}_i^{p,n} \boldsymbol{Y}^{p,n}(\boldsymbol{x}_i^p).$$

Collecting these vectors into the $n \times p_s$ matrix $\widetilde{\boldsymbol{U}}_{\text{nod},s}$,

$$\widetilde{\boldsymbol{U}}_{\text{nod},s} = \left[\widetilde{\boldsymbol{U}}_\text{nod}^s(\boldsymbol{x}_1^p)\ \widetilde{\boldsymbol{U}}_\text{nod}^s(\boldsymbol{x}_2^p) \cdots \widetilde{\boldsymbol{U}}_\text{nod}^s(\boldsymbol{x}_n^p)\right]^\top, \tag{8.4.61}$$

gives

$$\boldsymbol{\Lambda}_K^{-1} \boldsymbol{F}_K^\top \boldsymbol{P}^{p,n} \boldsymbol{Y}_{p,n} = \text{Vec}\left(\widetilde{\boldsymbol{U}}_{\text{nod},s}^\top\right). \tag{8.4.62}$$

Utilizing (8.4.62) and (B.5.4)

$$\boldsymbol{Q}_{p_s,n}\boldsymbol{\Lambda}_K^{-1}\boldsymbol{F}_K^\top\boldsymbol{P}^{p,n}\boldsymbol{Y}_{p,n} = \text{Vec}\left(\widetilde{\boldsymbol{U}}_{\text{nod},s}\right), \tag{8.4.63}$$

and therefore upon substituting (8.4.63) into (8.4.60)

$$\widehat{\boldsymbol{Z}}_{\text{nod}} = \begin{pmatrix}\text{Vec}\left(\widetilde{\boldsymbol{U}}_{\text{nod},s}\right)\\ \text{Vec}\left(\widetilde{\boldsymbol{W}}_s\right)\end{pmatrix}. \tag{8.4.64}$$

The $(n+m)p_s \times 1$ vector $\widehat{\boldsymbol{Z}}_{\text{nod}}$ does not depend on any unknown parameters. Substantial dimension reduction is achieved by using $\widehat{\boldsymbol{Z}}_{\text{nod}}$ as "data" in log likelihood calculations when $p_s \ll \min\{m_s, (m_y/n)\}$.

Leveraging (8.4.54), (8.4.27), (8.4.55), (8.4.29), (8.4.61), (8.4.46), and (8.4.64), along with (B.5.4) for the experimental data block and (B.5.5) for the simulator block, yields

$$(\boldsymbol{Y}-\boldsymbol{C}\widehat{\boldsymbol{\beta}})^\top\boldsymbol{\Sigma}_\varepsilon^{-1}(\boldsymbol{Y}-\boldsymbol{C}\widehat{\boldsymbol{\beta}}) = \lambda_{p,\varepsilon}\sum_{i=1}^{n}\left\|\boldsymbol{Y}^{p,n}(\boldsymbol{x}_i^p) - \boldsymbol{K}_i^s\widetilde{\boldsymbol{U}}_{\text{nod}}^s(\boldsymbol{x}_i^p)\right\|_{\boldsymbol{P}_i^{p,n}}^2$$
$$+\lambda_{s,\varepsilon}\sum_{i=1}^{m}\left\|\boldsymbol{Y}^{s,n,\epsilon}(\boldsymbol{x}_i^s,\boldsymbol{t}_i) - \boldsymbol{K}^s\widetilde{\boldsymbol{W}}^s(\boldsymbol{x}_i^s,\boldsymbol{t}_i)\right\|_{\boldsymbol{I}_{m_s}}^2. \tag{8.4.65}$$

Employing (8.4.56), $\boldsymbol{\Gamma} = diag(\lambda_{p,\varepsilon}r_0\boldsymbol{I}_{np_s}, \boldsymbol{0}_{mp_s,mp_s})$, and utilizing (8.4.64) produce

$$\widehat{\boldsymbol{\beta}}^\top\boldsymbol{\Gamma}\widehat{\boldsymbol{\beta}} = \lambda_{p,\varepsilon}\,r_0\sum_{i=1}^{n}\left\|\widetilde{\boldsymbol{U}}_{\text{nod}}^s(\boldsymbol{x}_i^p)\right\|_{\boldsymbol{I}_{p_s}}^2. \tag{8.4.66}$$

Higdon et al. (2008) neglected this term under the presumption that r_0 is small relative to the sum of the (estimated) squared coefficient norms. The sums in (8.4.65) and (8.4.66) do not depend on any unknown parameters, implying they can be computed once, stored, and accessed as needed.

Plugging (8.4.27), (8.4.58), (8.4.65), (8.4.66), (8.4.64), (8.4.53), and (8.4.59) into (8.4.35), the (approximate) log likelihood function of $\boldsymbol{\Omega}^{\text{nod, all}}$ is given by

$$\ell_{\text{nod}}(\boldsymbol{\Omega}^{\text{nod, all}};\boldsymbol{Y}_{p,n},\boldsymbol{Y}_{s,n,\varepsilon},r_0) =$$
$$\frac{m_y - np_s}{2}\,\ell n\left(\lambda_{p,\varepsilon}\right) + \frac{m(m_s-p_s)}{2}\,\ell n\left(\lambda_{s,\varepsilon}\right)$$
$$-\lambda_{p,\varepsilon}\left[\frac{1}{2}\sum_{i=1}^{n}\left\|\boldsymbol{Y}^{p,n}(\boldsymbol{x}_i^p) - \boldsymbol{K}_i^s\widetilde{\boldsymbol{U}}_{\text{nod}}^s(\boldsymbol{x}_i^p)\right\|_{\boldsymbol{P}_i^{p,n}}^2\right.$$
$$\left.+\frac{r_0}{2}\sum_{i=1}^{n}\left\|\widetilde{\boldsymbol{U}}_{\text{nod}}^s(\boldsymbol{x}_i^p)\right\|_{\boldsymbol{I}_{p_s}}^2\right]$$

$$- \lambda_{s,\varepsilon} \left[\frac{1}{2} \sum_{i=1}^{m} \left\| \boldsymbol{Y}^{s,n,\epsilon}(\boldsymbol{x}_i^s, \boldsymbol{t}_i) - \boldsymbol{K}^s \widetilde{\boldsymbol{W}}^s(\boldsymbol{x}_i^s, \boldsymbol{t}_i) \right\|_{\boldsymbol{I}_{m_s}}^2 \right]$$

$$- \frac{1}{2} \ell n \det\left(\boldsymbol{\Sigma}_{Z,\text{nod}} + \boldsymbol{\Sigma}_{\Gamma,\text{nod}}^{-1}\right) - \frac{1}{2} \widehat{\boldsymbol{Z}}_{\text{nod}}^{\top} \left(\boldsymbol{\Sigma}_{Z,\text{nod}} + \boldsymbol{\Sigma}_{\Gamma,\text{nod}}^{-1}\right)^{-1} \widehat{\boldsymbol{Z}}_{\text{nod}} . \quad (8.4.67)$$

Here $\boldsymbol{\Omega}^{\text{nod, all}} = \{\boldsymbol{\Omega}^{p,\text{nod},\varepsilon}, \lambda_{s,\varepsilon}\}$ is the collection of all unknown parameters appearing in the joint statistical model (8.4.54) and r_0 is defined as in (8.4.56). Additive constants not depending on $\boldsymbol{\Omega}^{\text{nod, all}}$ were excluded from (8.4.67) as they have no effect on statistical inference.

8.5 Bayesian Analysis

Section 8.5.1 specifies one possible prior for the parameters of the functional multivariate calibration model introduced in Sect. 8.4 and provides a justification of the prior. The final expressions for the log posteriors for the resulting discrepancy and no discrepancy scenarios are given by (8.5.32) and (8.5.33), respectively. Section 8.5.2 provides detailed methodology for using the posterior samples to implement the three emulation scenarios listed at the beginning of Sect. 8.4 and repeated here for convenience: at unsampled inputs

Case 1: emulate the simulation output using only simulator data
Case 2: emulate the calibrated simulator output modeling the simulator bias, and
Case 3: emulate the calibrated simulator output assuming no simulator bias.

8.5.1 Prior and Posterior Distributions

The log likelihood functions derived in Sect. 8.4.3 are combined with log prior distributions specified below to form the log posterior distributions used in Bayesian inference of all uncertain parameters for each modeling scenario previously described. This subsection concludes with a brief discussion of MCMC methods that have been used to sample these posterior distributions.

The specification of prior distributions given in the next paragraphs follows the approach taken in Higdon et al. (2008). Numerical values for prior parameters are suggested in this subsection, and their interpretation is discussed. *Alternative (parametric) prior distributions and choices of parameter values can certainly be used.*

First, the calibration parameter vector $\boldsymbol{\Theta}$ is assigned a prior distribution:

$$[\boldsymbol{\Theta}] \sim \pi(\boldsymbol{\theta}) , \quad (8.5.1)$$

where $\pi(\cdot)$ is a probability density function. Common choices for this distribution are uniform and multivariate normal. A uniform distribution allows the analyst to restrict each calibration parameter to a fixed, predefined interval or half interval. Intervals defined in terms of some fixed percentage deviation from a nominal value for each parameter are common in applications.

Example 1.7 *(Continued)* (*Cases 2 and 3: Emulation of the Calibrated Simulator*). The equation of state, spall strength, and impact velocity parameters ϵ, $P_{\min}$, and v_s are assigned independent uniform prior distributions on the domains provided in Table 1.4. The seven material strength parameters (with $-\ell n(\gamma)$ in place of γ) are assigned a multivariate normal prior distribution having means, standard deviations (SDs), and correlation matrix given in Table 8.8. This distribution was obtained by separately calibrating the Preston–Tonks–Wallace material strength model using data from a different set of experiments that isolated the strength behavior of tantalum. The prior distribution of the material strength parameters is truncated to their joint domain given in Table 1.4. ♦

	θ_0	κ	$-\ell n(\gamma)$	y_0	y_∞	s_0	s_∞
Mean	0.00818	0.713	11.1	0.0098	0.0016	0.0225	0.00351
SD/0.001	1.787	38.6	516.0	0.413	0.106	11.4	0.385
Correlation matrix							
θ_0	1	−0.14	−0.0394	−0.176	−0.874	−0.473	0.0093
κ	−0.14	1	0.726	0.607	0.26	0.102	0.207
$-\ell n(\gamma)$	−0.0394	0.726	1	0.0373	−0.0394	0.0829	0.0521
y_0	−0.176	0.607	0.0373	1	0.329	0.0104	0.312
y_∞	−0.874	0.26	−0.0394	0.329	1	0.323	0.114
s_0	−0.473	0.102	0.0829	0.0104	0.323	1	−0.66
s_∞	0.0093	0.207	0.0521	0.312	0.114	−0.66	1

Table 8.8 Prior distribution settings for material strength calibration parameters

Second, independent beta distributions are assigned to all correlation length parameters in Sects. 8.4.1 and 8.4.2, independent of $\boldsymbol{\Theta}$. In the *simulator basis model*, the priors for the control and calibration input correlation parameters are taken to be

$$[\rho_j^{x,i} \mid a_{s,\rho}^{ij}, b_{s,\rho}^{ij}] \sim Be(a_{s,\rho}^{ij}, b_{s,\rho}^{ij})\,;\ i = 1,\ldots,p_s\,,\ j = 1,\ldots,d\,, \tag{8.5.2}$$

$$[\rho_j^{t,i} \mid a_{s,\rho}^{ij}, b_{s,\rho}^{ij}] \sim Be(a_{s,\rho}^{ij}, b_{s,\rho}^{ij})\,;\ i = 1,\ldots,p_s\,,\ j = d+1,\ldots,d+q\,, \tag{8.5.3}$$

respectively, where the beta distribution $Be(\alpha,\beta)$ is parameterized as in (B.3.1) of Appendix B.3. In the *discrepancy basis model*, the prior

$$[\rho_j^{\delta,i} \,|\, a^{ij}_{\delta,\rho}, b^{ij}_{\delta,\rho}] \sim Be(a^{ij}_{\delta,\rho}, b^{ij}_{\delta,\rho})\,;\; i = 1,\ldots,F\,,\; j = 1,\ldots,d\,, \tag{8.5.4}$$

is assumed.

Although the values of the fixed prior parameters are allowed to vary across correlation length parameters, as indicated by the notation above, they will each typically be set to common values $a_{s,\rho}$ and $b_{s,\rho}$ in the simulator basis model and $a_{\delta,\rho}$ and $b_{\delta,\rho}$ in the discrepancy basis model. This same reasoning is also used to simplify the selection of prior parameters in other prior parameter settings introduced below. In many applications, the principle of *effect sparsity* is reasonable to assume, i.e., the output variation can be assumed to be dominated by only a few input variables. In the absence of specific knowledge relevant to a particular application, this principle will be enforced by choice of appropriate fixed parameter settings for these beta prior distributions. If any correlation length parameter assumes the value 1, variation of its corresponding input variable has no impact on the associated GP. Therefore, effect sparsity can be enforced by adopting beta prior distributions having means close to 1 with "small" variances. For example, setting $(a_{s,\rho}, b_{s,\rho}) = (1, 0.1) = (a_{\delta,\rho}, b_{\delta,\rho})$ defines a beta distribution having approximately 75% of its probability mass above 0.95.

Third, the prior for all precision parameters encountered in Sects. 8.4.1 and 8.4.2 is assumed independent of all correlation length parameters and $\boldsymbol{\Theta}$; the precision parameters are assumed to have independent gamma distributions. For the simulator basis model

$$[\lambda_{s,i} \,|\, a^i_{s,\lambda}] \sim \Gamma(a^i_{s,\lambda}, a^i_{s,\lambda})\,;\; i = 1,\ldots,p_s\,, \text{ and} \tag{8.5.5}$$

$$[\lambda_{n,i} \,|\, a^i_{n,\lambda}, b^i_{n,\lambda}] \sim \Gamma(a^i_{n,\lambda}, b^i_{n,\lambda})\,;\; i = 1,\ldots,p_s\,, \tag{8.5.6}$$

where the gamma distribution $\Gamma(\alpha,\beta)$ is parameterized as in (B.2.1) of Appendix B.2. Notice that the gamma prior distributions of $\{\lambda_{s,i}\}_{i=1}^{p_s}$ depend on the *single fixed parameter* $a_{s,\lambda}$; this selection of equal a and b parameters causes these prior distributions to have mean 1, implying that the simulator basis coefficient processes all have variance 1 conditionally when their precision parameters, $\lambda_{s,i}$, equal their prior means. This is consistent with the effect of simulation output standardization forcing a variance of 1 on the observed simulator basis coefficients as discussed in Sect. 8.4.1. Larger values of $a_{s,\lambda}$ force the prior precisions of the simulator basis coefficient processes to deviate less from 1; for example, when $a_{s,\lambda} = 5$, approximately 75% of the prior probability mass falls in [0.5, 1.5], while for $a_{s,\lambda} = 10$, approximately 90% of the prior probability mass falls in [0.5, 1.5]. The gamma prior for each $\lambda_{s,i}$ is truncated below at 0.3 to prevent excessively large simulator basis coefficient process variances.

The gamma prior distributions for the nugget parameter precisions, $\{\lambda_{n,i}\}_{i=1}^{p_s}$, are chosen so that the mean nugget effects are "small," but have large variances to allow any of these effects to activate as necessary for numerical stability. For example, $(a_{n,\lambda}, b_{n,\lambda}) = (3, 0.003)$ accomplishes this goal, yielding a "large" prior mean and prior variance for $\lambda_{n,i}$ (recall a large precision is a small variance, so that a mean-zero nugget effect process having a large precision parameter will take values near zero with high probability). The gamma prior for each $\lambda_{n,i}$ is truncated below at 60 to prevent the nugget effects from getting too "large."

For the *discrepancy basis model* that allows different biases in F portions of the input space, the prior

$$[\lambda_{\delta,i} \mid a^i_{\delta,\lambda}, b^i_{\delta,\lambda}] \sim \Gamma(a^i_{\delta,\lambda}, b^i_{\delta,\lambda}) \,,\; i = 1, \ldots, F \tag{8.5.7}$$

is assumed. Initially it is common to assume that the simulator is a good representation of physical reality, meaning the discrepancy should be close to zero across the input domain; nonzero discrepancy $\delta(\boldsymbol{x})$ should be allowed to activate if this assumption is incorrect, even in a part of the input domain. Both objectives are accomplished by establishing gamma prior distributions on the precision parameters of the discrepancy basis coefficient processes that have large means and variances, using the same reasoning as discussed in the context of the nugget effects above. The choice $(a_{\delta,\lambda}, b_{\delta,\lambda}) = (1, 0.001)$ enforces, initially, a universally small discrepancy process with substantial flexibility for targeted activation through selection of $F > 1$ coefficient groups, if necessary.

Finally, for the observational error of the physical experiment and the simulator basis noise

$$[\lambda_{p,\varepsilon} \mid a_{p,\varepsilon}, b_{p,\varepsilon}] \sim \Gamma\left(a_{p,\varepsilon}, b_{p,\varepsilon}\right) \,,\; \text{independent of}$$

$$[\lambda_{s,\varepsilon} \mid a_{s,\varepsilon}, b_{s,\varepsilon}] \sim \Gamma\left(a_{s,\varepsilon}, b_{s,\varepsilon}\right)$$

(and both independent of all other parameters). The prior distribution for observational error precision is often chosen to be diffuse (having a "large" variance) when replicate physical experiments have been conducted; the prior pair $(a_{p,\varepsilon}, b_{p,\varepsilon}) = (1, 0.001)$ permits such behavior. This $(a_{p,\varepsilon}, b_{p,\varepsilon})$ choice allows the actual variability in the replicates to inform this precision with minimal restriction. On the other hand, when physical observations are scarce, the prior distribution of this precision is typically chosen to be informative (having a "small" variance) with its probability mass focused on an assumed level of observational error derived from subject-matter expert assessment.

The simulator basis noise should typically be small if a sufficient number of simulator basis vectors, p_s, are chosen to represent simulation output. Therefore the prior mean precision of the mean-zero simulator basis noise process is set to be "large" as is the prior variance of this precision, to allow for activation of this process in settings where greater simulator basis noise is present; for example, $(a_{s,\varepsilon}, b_{s,\varepsilon}) = (5, 0.005)$ has these two features. However, the gamma prior of $\lambda_{s,\varepsilon}$ is truncated below at 60 to prevent simulator basis noise from becoming too "large."

Example 1.7 *(Continued)*. The behavior of the simulator and discrepancy basis coefficient processes (8.4.6) and (8.4.17) is governed by the correlation length parameters (8.5.2), (8.5.3), and (8.5.4), as well as the precision parameters (8.5.5), (8.5.6), and (8.5.7). Table 8.9 provides the settings of the fixed parameters defining the prior distributions of these correlation length and precision parameters. These choices are common across input dimensions (for the correlation length parameters) and coefficient processes. In addition, the settings of the fixed parameters defining the prior distributions of the simulator basis noise precision $\lambda_{s,\varepsilon}$ and observation error precision $\lambda_{p,\varepsilon}$ are also stated. The choices made for all of these parameters are seen to be in line with the above discussion. Finally, it is noted that this analysis assumes an $F = 1$ parameter group for the discrepancy basis coefficient processes (8.4.17). ♦

$a_{s,\rho}$	$b_{s,\rho}$	$a_{s,\lambda}$	$a_{n,\lambda}$	$b_{n,\lambda}$	$a_{\delta,\rho}$	$b_{\delta,\rho}$	$a_{\delta,\lambda}$	$b_{\delta,\lambda}$	$a_{s,\varepsilon}$	$b_{s,\varepsilon}$	$a_{p,\varepsilon}$	$b_{p,\varepsilon}$
1	0.1	5	3	0.003	1	0.1	1	0.001	5	0.005	1	0.001

Table 8.9 Prior distribution settings for correlation length and precision parameters

Because the log posterior distribution is, by definition, the sum of the log prior distribution and the log likelihood function, it is evident from the form of the log likelihood functions (8.4.51) and (8.4.67) that the leading terms involving $\lambda_{s,\varepsilon}$ and $\lambda_{p,\varepsilon}$ from these log likelihood functions can be grouped with the $\lambda_{s,\varepsilon}$ and $\lambda_{p,\varepsilon}$ terms from the log prior distributions yielding an equivalent expression for the log posterior distribution. Specifically, define the following constants:

$$a_{\mathrm{d},p,\varepsilon} = a_{p,\varepsilon} + \frac{m_y - n(p_\delta + p_s)}{2},$$

$$b_{\mathrm{d},p,\varepsilon} = b_{p,\varepsilon} + \frac{1}{2}\sum_{i=1}^{n} \left\| \boldsymbol{Y}^{p,n}(\boldsymbol{x}_i^p) - \boldsymbol{D}_i^p \widetilde{\boldsymbol{V}}^p(\boldsymbol{x}_i^p) - \boldsymbol{K}_i^s \widetilde{\boldsymbol{U}}_{\mathrm{d}}^s(\boldsymbol{x}_i^p) \right\|^2_{\boldsymbol{P}_i^{p,n}}$$

$$+ \frac{r_0}{2}\sum_{i=1}^{n} \left(\left\| \widetilde{\boldsymbol{V}}^p(\boldsymbol{x}_i^p) \right\|^2_{\boldsymbol{I}_{p_\delta}} + \left\| \widetilde{\boldsymbol{U}}_{\mathrm{d}}^s(\boldsymbol{x}_i^p) \right\|^2_{\boldsymbol{I}_{p_s}} \right), \tag{8.5.8}$$

$$a_{\mathrm{nod},p,\varepsilon} = a_{p,\varepsilon} + \frac{m_y - n p_s}{2},$$

$$b_{\mathrm{nod},p,\varepsilon} = b_{p,\varepsilon} + \frac{1}{2}\sum_{i=1}^{n} \left\| \boldsymbol{Y}^{p,n}(\boldsymbol{x}_i^p) - \boldsymbol{K}_i^s \widetilde{\boldsymbol{U}}_{\mathrm{nod}}^s(\boldsymbol{x}_i^p) \right\|^2_{\boldsymbol{P}_i^{p,n}}$$

$$+ \frac{r_0}{2}\sum_{i=1}^{n} \left\| \widetilde{\boldsymbol{U}}_{\mathrm{nod}}^s(\boldsymbol{x}_i^p) \right\|^2_{\boldsymbol{I}_{p_s}}, \tag{8.5.9}$$

$$a'_{s,\varepsilon} = a_{s,\varepsilon} + \frac{m(m_s - p_s)}{2},$$

$$b'_{s,\varepsilon} = b_{s,\varepsilon} + \frac{1}{2}\sum_{i=1}^{m} \left\| \boldsymbol{Y}^{s,n,\epsilon}(\boldsymbol{x}_i^s, \boldsymbol{t}_i) - \boldsymbol{K}^s \widetilde{\boldsymbol{W}}^s(\boldsymbol{x}_i^s, \boldsymbol{t}_i) \right\|^2_{\boldsymbol{I}_{m_s}},$$

where r_0 is defined as in (8.4.37) or (8.4.56). The modified prior distributions of $\lambda_{p,\varepsilon}$ with and without modeling of simulator discrepancy become

$$[\lambda_{p,\varepsilon} \mid a_{\mathrm{d},p,\varepsilon}, b_{\mathrm{d},p,\varepsilon}] \sim \Gamma\left(a_{\mathrm{d},p,\varepsilon}, b_{\mathrm{d},p,\varepsilon}\right)\,, \quad \text{and} \tag{8.5.10}$$

$$[\lambda_{p,\varepsilon} \mid a_{\mathrm{nod},p,\varepsilon}, b_{\mathrm{nod},p,\varepsilon}] \sim \Gamma\left(a_{\mathrm{nod},p,\varepsilon}, b_{\mathrm{nod},p,\varepsilon}\right)\,, \tag{8.5.11}$$

respectively. In particular, when discrepancy is modeled, $a_{p,\varepsilon} > 0$ must take a value sufficiently large so that $a_{\mathrm{d},p,\varepsilon} > 0$ is satisfied. This is automatic for any $a_{p,\varepsilon} > 0$ when $r_0 = 0$ is an allowable choice. The modified prior distribution of $\lambda_{s,\varepsilon}$ takes the form

$$[\lambda_{s,\varepsilon} \mid a'_{s,\varepsilon}, b'_{s,\varepsilon}] \sim \Gamma\left(a'_{s,\varepsilon}, b'_{s,\varepsilon}\right)\,. \tag{8.5.12}$$

Example 1.7 *(Continued)*. Table 8.10 lists the settings of the parameters defining the $\lambda_{p,\varepsilon}$ prior distributions when the simulator discrepancy is modeled (8.5.10) and when the simulator discrepancy is omitted (8.5.11). Observe that with high probability the modified prior precision $\lambda_{p,\varepsilon}$ is *larger* when the simulator discrepancy is modeled than when it is omitted (prior mean of 14.8 and standard deviation of 1.66 versus 2.078 and 0.226). This is a consequence of taking $r_0 = 0$ in this analysis and a smaller residual sum of squares (between the experimental data and its least squares estimates) observed in Fig. 8.18 when compared with Fig. 8.19. These residual sums of squares appear in (8.5.8) and (8.5.9), with the consequence that $b_{\mathrm{d},p,\varepsilon} \ll b_{\mathrm{nod},p,\varepsilon}$ in this analysis. The result is a smaller modified prior error variance when simulator discrepancy is modeled than when it is omitted. The settings of the modified prior distribution (8.5.12) of $\lambda_{s,\varepsilon}$ are also given in Table 8.10. The amount of simulation data that is available results in this distribution having a very sharp peak at its mean value 21.7.

$a_{\mathrm{d},p,\varepsilon}$	$b_{\mathrm{d},p,\varepsilon}$	$a_{\mathrm{nod},p,\varepsilon}$	$b_{\mathrm{nod},p,\varepsilon}$	$a'_{s,\varepsilon}$	$b'_{s,\varepsilon}$
80	5.4	84.5	40.7	8520	392

Table 8.10 Modified prior distribution settings for $\lambda_{p,\varepsilon}$ and $\lambda_{s,\varepsilon}$ under Cases 2 and 3

In the case of emulating the simulation output based only on the 127 simulator runs, the modified prior distribution of $\lambda_{s,\varepsilon}$ is slightly different because of the removal of a simulation run: $a'_{s,\varepsilon} = 8450$ and $b'_{s,\varepsilon} = 393$.

Recall from Table 8.9 that the initial gamma priors for $\lambda_{s,\varepsilon}$ and $\lambda_{p,\varepsilon}$ were chosen to be diffuse. As seen in Table 8.10, the modified priors for these parameters allow the size of the simulator and experimental data model residuals to influence the size of the simulator basis noise and observational error processes. In this example, only $n = 1$ experiment is available for calibration. Therefore, care should be exercised to ensure simulator bias is not overfit by the discrepancy model, due to its considerable impact on the observational error process through the modified prior for $\lambda_{p,\varepsilon}$. Ideally, replicate experiments would be available to provide a pure estimate of observational error through this modified prior distribution. ♦

Stated in terms of arbitrary given prior parameters, the log prior distribution of $\boldsymbol{\Omega}^{\mathrm{d,\,all}}$ (where the discrepancy is modeled statistically) is obtained by collecting (8.5.1), (8.5.2), (8.5.3), (8.5.4), (8.5.5), (8.5.6), (8.5.7), (8.5.10), and (8.5.12) yielding

$$\begin{aligned}
p_{\text{d}}(\boldsymbol{\Omega}^{\text{d, all}}; r_0) = \ell n\,(\pi(\boldsymbol{\theta}))
&+ \sum_{i=1}^{p_s}\sum_{j=1}^{d}\Big((a_{s,\rho}^{ij}-1)\,\ell n(\rho_j^{x,i}) + (b_{s,\rho}^{ij}-1)\,\ell n(1-\rho_j^{x,i})\Big)\\
&+ \sum_{i=1}^{p_s}\sum_{j=d+1}^{d+q}\Big((a_{s,\rho}^{ij}-1)\,\ell n(\rho_j^{t,i}) + (b_{s,\rho}^{ij}-1)\,\ell n(1-\rho_j^{t,i})\Big)\\
&+ \sum_{i=1}^{F}\sum_{j=1}^{d}\Big((a_{\delta,\rho}^{ij}-1)\,\ell n(\rho_j^{\delta,i}) + (b_{\delta,\rho}^{ij}-1)\,\ell n(1-\rho_j^{\delta,i})\Big)\\
&+ \sum_{i=1}^{p_s}\Big((a_{s,\lambda}^{i}-1)\,\ell n(\lambda_{s,i}) - a_{s,\lambda}^{i}\lambda_{s,i}\Big)\\
&+ \sum_{i=1}^{p_s}\Big((a_{n,\lambda}^{i}-1)\,\ell n(\lambda_{n,i}) - b_{n,\lambda}^{i}\lambda_{n,i}\Big)\\
&+ \sum_{i=1}^{F}\Big((a_{\delta,\lambda}^{i}-1)\,\ell n(\lambda_{\delta,i}) - b_{\delta,\lambda}^{i}\lambda_{\delta,i}\Big)\\
&+ (a_{\text{d},p,\varepsilon}-1)\,\ell n(\lambda_{p,\varepsilon}) - b_{\text{d},p,\varepsilon}\lambda_{p,\varepsilon}\\
&+ (a'_{s,\varepsilon}-1)\,\ell n(\lambda_{s,\varepsilon}) - b'_{s,\varepsilon}\lambda_{s,\varepsilon}\,, \qquad (8.5.13)
\end{aligned}$$

up to additive terms that do not change as a function of $\boldsymbol{\Omega}^{\text{d, all}}$.

Analogously, the log prior distribution of $\boldsymbol{\Omega}^{\text{nod, all}}$ (where the discrepancy is assumed to be negligible) is obtained by collecting (8.5.1), (8.5.2), (8.5.3), (8.5.5), (8.5.6), (8.5.11), and (8.5.12) producing

$$\begin{aligned}
p_{\text{nod}}(\boldsymbol{\Omega}^{\text{nod, all}}; r_0) = \ell n\,(\pi(\boldsymbol{\theta}))
&+ \sum_{i=1}^{p_s}\sum_{j=1}^{d}\Big((a_{s,\rho}^{ij}-1)\,\ell n(\rho_j^{x,i}) + (b_{s,\rho}^{ij}-1)\,\ell n(1-\rho_j^{x,i})\Big)\\
&+ \sum_{i=1}^{p_s}\sum_{j=d+1}^{d+q}\Big((a_{s,\rho}^{ij}-1)\,\ell n(\rho_j^{t,i}) + (b_{s,\rho}^{ij}-1)\,\ell n(1-\rho_j^{t,i})\Big)\\
&+ \sum_{i=1}^{p_s}\Big((a_{s,\lambda}^{i}-1)\,\ell n(\lambda_{s,i}) - a_{s,\lambda}^{i}\lambda_{s,i}\Big)\\
&+ \sum_{i=1}^{p_s}\Big((a_{n,\lambda}^{i}-1)\,\ell n(\lambda_{n,i}) - b_{n,\lambda}^{i}\lambda_{n,i}\Big)
\end{aligned}$$

$$+ (a_{\mathrm{nod},p,\varepsilon} - 1)\, \ell n(\lambda_{p,\varepsilon}) - b_{\mathrm{nod},p,\varepsilon}\lambda_{p,\varepsilon}$$
$$+ (a'_{s,\varepsilon} - 1)\, \ell n(\lambda_{s,\varepsilon}) - b'_{s,\varepsilon}\lambda_{s,\varepsilon}\,, \tag{8.5.14}$$

up to additive terms that do not change as a function of $\boldsymbol{\Omega}^{\mathrm{nod,\,all}}$.

With the modified prior distribution (8.5.10) for the observation error and (8.5.12) for the simulator basis noise incorporated into (8.5.13), the log likelihood function (8.4.51) reduces to

$$\ell_{\mathrm{d}}(\boldsymbol{\Omega}^{\mathrm{d,\,all}}; \widehat{\boldsymbol{Z}}_{\mathrm{d}}, r_0) = -\frac{1}{2}\,\ell n \det\left(\boldsymbol{\Sigma}_{Z,\mathrm{d}} + \boldsymbol{\Sigma}^{-1}_{\Gamma,\mathrm{d}}\right) - \frac{1}{2}\widehat{\boldsymbol{Z}}^{\top}_{\mathrm{d}}\left(\boldsymbol{\Sigma}_{Z,\mathrm{d}} + \boldsymbol{\Sigma}^{-1}_{\Gamma,\mathrm{d}}\right)^{-1}\widehat{\boldsymbol{Z}}_{\mathrm{d}}\,, \tag{8.5.15}$$

while (8.4.67) takes a similarly reduced form after incorporation of the modified prior distribution (8.5.11) for observation error and (8.5.12) for simulator basis noise into (8.5.14)

$$\ell_{\mathrm{nod}}(\boldsymbol{\Omega}^{\mathrm{nod,\,all}}; \widehat{\boldsymbol{Z}}_{\mathrm{nod}}, r_0) =$$
$$-\frac{1}{2}\,\ell n \det\left(\boldsymbol{\Sigma}_{Z,\mathrm{nod}} + \boldsymbol{\Sigma}^{-1}_{\Gamma,\mathrm{nod}}\right) - \frac{1}{2}\widehat{\boldsymbol{Z}}^{\top}_{\mathrm{nod}}\left(\boldsymbol{\Sigma}_{Z,\mathrm{nod}} + \boldsymbol{\Sigma}^{-1}_{\Gamma,\mathrm{nod}}\right)^{-1}\widehat{\boldsymbol{Z}}_{\mathrm{nod}}\,. \tag{8.5.16}$$

If it is feasible to take $r_0 = 0$ in (8.4.37) or (8.4.56), then (8.5.15) and (8.5.16) are the log likelihood functions of reduced data $\widehat{\boldsymbol{Z}}_{\mathrm{d}}$ and $\widehat{\boldsymbol{Z}}_{\mathrm{nod}}$, respectively, as discussed in Sect. 8.4.3 immediately following (8.4.36). When $r_0 > 0$, this notation is maintained even though (8.5.15) and (8.5.16) only approximate the actual log likelihood functions of the reduced data in this case.

By providing alternative expressions for the matrices $\boldsymbol{\Sigma}_{Z,\mathrm{d}} + \boldsymbol{\Sigma}^{-1}_{\Gamma,\mathrm{d}}$ and $\boldsymbol{\Sigma}_{Z,\mathrm{nod}} + \boldsymbol{\Sigma}^{-1}_{\Gamma,\mathrm{nod}}$, the log likelihood functions (8.5.15) and (8.5.16) can be expressed in such a way as to provide important computational improvements. These equivalent expressions use the following matrix algebra identities. Consider the (generic) symmetric matrix $\boldsymbol{A}$ partitioned as follows:

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}^{\top}_{12} & \boldsymbol{A}_{22} \end{pmatrix}.$$

If both $\boldsymbol{A}_{22}$ and $\boldsymbol{S}_{11} = \boldsymbol{A}_{11} - \boldsymbol{A}_{12}\boldsymbol{A}^{-1}_{22}\boldsymbol{A}^{\top}_{12}$ are nonsingular, then Lemma B.3 gives

$$\boldsymbol{A}^{-1} = \begin{pmatrix} \boldsymbol{S}^{-1}_{11} & -\boldsymbol{S}^{-1}_{11}\boldsymbol{A}_{12}\boldsymbol{A}^{-1}_{22} \\ -\boldsymbol{A}^{-1}_{22}\boldsymbol{A}^{\top}_{12}\boldsymbol{S}^{-1}_{11} & \boldsymbol{A}^{-1}_{22} + \boldsymbol{A}^{-1}_{22}\boldsymbol{A}^{\top}_{12}\boldsymbol{S}^{-1}_{11}\boldsymbol{A}_{12}\boldsymbol{A}^{-1}_{22} \end{pmatrix} \tag{8.5.17}$$

and the determinant of $\boldsymbol{A}$ is can be computed as

$$\det(\boldsymbol{A}) = \det(\boldsymbol{S}_{11})\,\det(\boldsymbol{A}_{22})\,. \tag{8.5.18}$$

When the simulator discrepancy is modeled statistically, take the $\boldsymbol{A}$ block entries for $\boldsymbol{\Sigma}_{Z,\mathrm{d}} + \boldsymbol{\Sigma}^{-1}_{\Gamma,\mathrm{d}}$ from (8.4.23) and (8.4.40) to be

$$\boldsymbol{A}_{11} = \begin{pmatrix} \boldsymbol{\Sigma}_V & \boldsymbol{0}_{np_\delta,np_s} \\ \boldsymbol{0}_{np_s,np_\delta} & \boldsymbol{\Sigma}_U \end{pmatrix} + \lambda_{p,\varepsilon}^{-1}\boldsymbol{Q}_{p_\delta+p_s,n}\boldsymbol{\Lambda}_{DK}^{-1}\boldsymbol{Q}_{n,p_\delta+p_s}\,,\ \boldsymbol{A}_{12} = \begin{pmatrix} \boldsymbol{0}_{np_\delta,mp_s} \\ \boldsymbol{\Sigma}_{UW} \end{pmatrix},$$

and $\boldsymbol{A}_{22} = \boldsymbol{\Sigma}_{W\varepsilon}$ where

$$\boldsymbol{\Sigma}_{W\varepsilon} = diag\left(\boldsymbol{\Sigma}_{W\varepsilon,1}, \boldsymbol{\Sigma}_{W\varepsilon,2}, \ldots, \boldsymbol{\Sigma}_{W\varepsilon,p_s}\right) \tag{8.5.19}$$

has diagonal blocks

$$\boldsymbol{\Sigma}_{W\varepsilon,i} = \boldsymbol{\Sigma}_{W,i} + \lambda_{s,\varepsilon}^{-1} m \left(\widetilde{\sigma}_{i,i}^{s}\right)^{-2} \boldsymbol{I}_m\,,\ i = 1, \ldots, p_s\,, \tag{8.5.20}$$

for $\widetilde{\boldsymbol{\Sigma}}_{11}^{s} = diag\left(\widetilde{\sigma}_{1,1}^{s}, \widetilde{\sigma}_{2,2}^{s}, \ldots, \widetilde{\sigma}_{p_s,p_s}^{s}\right)$. Define

$$\boldsymbol{\Sigma}_{U|W} = \boldsymbol{\Sigma}_U - \boldsymbol{\Sigma}_{UW}\boldsymbol{\Sigma}_{W\varepsilon}^{-1}\boldsymbol{\Sigma}_{UW}^{\top}\,, \tag{8.5.21}$$

yielding

$$\boldsymbol{S}_{11} = \boldsymbol{\Sigma}_{\mathrm{d}|W} = \begin{pmatrix} \boldsymbol{\Sigma}_V & \boldsymbol{0}_{np_\delta,np_s} \\ \boldsymbol{0}_{np_s,np_\delta} & \boldsymbol{\Sigma}_{U|W} \end{pmatrix} + \lambda_{p,\varepsilon}^{-1}\boldsymbol{Q}_{p_\delta+p_s,n}\boldsymbol{\Lambda}_{DK}^{-1}\boldsymbol{Q}_{n,p_\delta+p_s} \tag{8.5.22}$$

in (8.5.17). Consider the $n \times p_s$ matrix $\widetilde{\boldsymbol{T}}_{\mathrm{d},s}$:

$$\widetilde{\boldsymbol{T}}_{\mathrm{d},s} = \left[\widetilde{\boldsymbol{U}}_{\mathrm{d},s}^{1} - \boldsymbol{\Sigma}_{UW,1}\boldsymbol{\Sigma}_{W\varepsilon,1}^{-1}\widetilde{\boldsymbol{W}}_{s}^{1} \cdots \widetilde{\boldsymbol{U}}_{\mathrm{d},s}^{p_s} - \boldsymbol{\Sigma}_{UW,p_s}\boldsymbol{\Sigma}_{W\varepsilon,p_s}^{-1}\widetilde{\boldsymbol{W}}_{s}^{p_s}\right], \tag{8.5.23}$$

where $\widetilde{\boldsymbol{U}}_{\mathrm{d},s}^{i}$ and $\widetilde{\boldsymbol{W}}_{s}^{i}$ are the i^{th} columns of the matrices $\widetilde{\boldsymbol{U}}_{\mathrm{d},s}$ from (8.4.43) and $\widetilde{\boldsymbol{W}}_{s}$ from (8.4.46), respectively, for $i = 1, \ldots, p_s$. Utilizing (8.5.23), (8.5.22), and (8.5.20) gives

$$\widehat{\boldsymbol{Z}}_{\mathrm{d}}^{\top}\left(\boldsymbol{\Sigma}_{Z,\mathrm{d}} + \boldsymbol{\Sigma}_{\Gamma,\mathrm{d}}^{-1}\right)^{-1}\widehat{\boldsymbol{Z}}_{\mathrm{d}} =$$

$$\mathrm{Vec}\left(\left[\widetilde{\boldsymbol{V}}_p\ \widetilde{\boldsymbol{T}}_{\mathrm{d},s}\right]\right)^{\top}\boldsymbol{\Sigma}_{\mathrm{d}|W}^{-1}\mathrm{Vec}\left(\left[\widetilde{\boldsymbol{V}}_p\ \widetilde{\boldsymbol{T}}_{\mathrm{d},s}\right]\right) + \sum_{i=1}^{p_s}\left(\widetilde{\boldsymbol{W}}_{s}^{i}\right)^{\top}\boldsymbol{\Sigma}_{W\varepsilon,i}^{-1}\widetilde{\boldsymbol{W}}_{s}^{i}\,. \tag{8.5.24}$$

The determinant formula (8.5.18) yields

$$\det\left(\boldsymbol{\Sigma}_{Z,\mathrm{d}} + \boldsymbol{\Sigma}_{\Gamma,\mathrm{d}}^{-1}\right) = \det\left(\boldsymbol{\Sigma}_{\mathrm{d}|W}\right)\,\det\left(\boldsymbol{\Sigma}_{W\varepsilon}\right) = \det\left(\boldsymbol{\Sigma}_{\mathrm{d}|W}\right)\prod_{i=1}^{p_s}\det\left(\boldsymbol{\Sigma}_{W\varepsilon,i}\right)\,. \tag{8.5.25}$$

Therefore, using (8.5.24) and (8.5.25), the log likelihood function (8.5.15) is equivalently given by

$$\ell_{\mathrm{d}}(\boldsymbol{\Omega}^{\mathrm{d,\,all}}; \widehat{\boldsymbol{Z}}_{\mathrm{d}}, r_0) =$$

$$-\frac{1}{2}\,\ell n\det\left(\boldsymbol{\Sigma}_{\mathrm{d}|W}\right) - \frac{1}{2}\mathrm{Vec}\left(\left[\widetilde{\boldsymbol{V}}_p\ \widetilde{\boldsymbol{T}}_{\mathrm{d},s}\right]\right)^{\top}\boldsymbol{\Sigma}_{\mathrm{d}|W}^{-1}\mathrm{Vec}\left(\left[\widetilde{\boldsymbol{V}}_p\ \widetilde{\boldsymbol{T}}_{\mathrm{d},s}\right]\right)$$

$$-\frac{1}{2}\sum_{i=1}^{p_s} \ell n \det\left(\boldsymbol{\Sigma}_{W\varepsilon,i}\right) - \frac{1}{2}\sum_{i=1}^{p_s}\left(\widetilde{\boldsymbol{W}}_s^i\right)^\top \boldsymbol{\Sigma}_{W\varepsilon,i}^{-1}\widetilde{\boldsymbol{W}}_s^i . \tag{8.5.26}$$

The computational complexity of (8.5.26) for repeated evaluations at different parameter settings $\boldsymbol{\Omega}^{\text{d, all}}$ (excluding one-time calculations of quantities that can be stored and accessed as needed) is $O\left((p_\delta + p_s)^3 n^3 + p_s m^3\right)$, while evaluation of the log likelihood function without the dimension-reducing simplifications is of the order $O\left((m_y + m_s m)^3\right)$. With $p_s \ll m_s$ and $(p_\delta + p_s) \ll (m_y/n)$ as assumed previously, (8.5.26) therefore achieves a significant computational cost savings.

When the simulator discrepancy is assumed to be identically zero, take the $\boldsymbol{A}$ block entries for $\boldsymbol{\Sigma}_{Z,\text{nod}} + \boldsymbol{\Sigma}_{\Gamma,\text{nod}}^{-1}$ from (8.4.53) and (8.4.59) to be

$$\boldsymbol{A}_{11} = \boldsymbol{\Sigma}_U + \lambda_{p,\varepsilon}^{-1}\boldsymbol{Q}_{p_s,n}\boldsymbol{\Lambda}_K^{-1}\boldsymbol{Q}_{n,p_s}\,,\ \boldsymbol{A}_{12} = \boldsymbol{\Sigma}_{UW}\,,\ \text{and } \boldsymbol{A}_{22} = \boldsymbol{\Sigma}_{W\varepsilon}\,.$$

Using $\boldsymbol{\Sigma}_{U|W}$ from (8.5.21), it is straightforward to calculate

$$\boldsymbol{S}_{11} = \boldsymbol{\Sigma}_{\text{nod}|W} = \boldsymbol{\Sigma}_{U|W} + \lambda_{p,\varepsilon}^{-1}\boldsymbol{Q}_{p_s,n}\boldsymbol{\Lambda}_K^{-1}\boldsymbol{Q}_{n,p_s} \tag{8.5.27}$$

in (8.5.17). Denote the $n \times p_s$ matrix $\widetilde{\boldsymbol{T}}_{\text{nod},s}$ to be

$$\widetilde{\boldsymbol{T}}_{\text{nod},s} = \left[\widetilde{\boldsymbol{U}}_{\text{nod},s}^1 - \boldsymbol{\Sigma}_{UW,1}\boldsymbol{\Sigma}_{W\varepsilon,1}^{-1}\widetilde{\boldsymbol{W}}_s^1 \cdots \widetilde{\boldsymbol{U}}_{\text{nod},s}^{p_s} - \boldsymbol{\Sigma}_{UW,p_s}\boldsymbol{\Sigma}_{W\varepsilon,p_s}^{-1}\widetilde{\boldsymbol{W}}_s^{p_s}\right], \tag{8.5.28}$$

where $\widetilde{\boldsymbol{U}}_{\text{nod},s}^i$ is the i^{th} column of the matrix $\widetilde{\boldsymbol{U}}_{\text{nod},s}$ from (8.4.61) for $i = 1, \ldots, p_s$. Utilizing (8.5.28), (8.5.27), and (8.5.20) gives

$$\widehat{\boldsymbol{Z}}_{\text{nod}}^\top\left(\boldsymbol{\Sigma}_{Z,\text{nod}} + \boldsymbol{\Sigma}_{\Gamma,\text{nod}}^{-1}\right)^{-1}\widehat{\boldsymbol{Z}}_{\text{nod}} =$$

$$\text{Vec}\left(\widetilde{\boldsymbol{T}}_{\text{nod},s}\right)^\top \boldsymbol{\Sigma}_{\text{nod}|W}^{-1}\text{Vec}\left(\widetilde{\boldsymbol{T}}_{\text{nod},s}\right) + \sum_{i=1}^{p_s}\left(\widetilde{\boldsymbol{W}}_s^i\right)^\top \boldsymbol{\Sigma}_{W\varepsilon,i}^{-1}\widetilde{\boldsymbol{W}}_s^i . \tag{8.5.29}$$

Apply (8.5.18) to show

$$\det\left(\boldsymbol{\Sigma}_{Z,\text{nod}} + \boldsymbol{\Sigma}_{\Gamma,\text{nod}}^{-1}\right) = \det\left(\boldsymbol{\Sigma}_{\text{nod}|W}\right)\,\det\left(\boldsymbol{\Sigma}_{W\varepsilon}\right) = \det\left(\boldsymbol{\Sigma}_{\text{nod}|W}\right)\prod_{i=1}^{p_s}\det\left(\boldsymbol{\Sigma}_{W\varepsilon,i}\right). \tag{8.5.30}$$

Therefore, using (8.5.29) and (8.5.30), the log likelihood function (8.5.16) is given equivalently by

$$\ell_{\text{nod}}(\boldsymbol{\Omega}^{\text{nod, all}}; \widehat{\boldsymbol{Z}}_{\text{nod}}, r_0) =$$

$$-\frac{1}{2}\ell n\det\left(\boldsymbol{\Sigma}_{\text{nod}|W}\right) - \frac{1}{2}\text{Vec}\left(\widetilde{\boldsymbol{T}}_{\text{nod},s}\right)^\top \boldsymbol{\Sigma}_{\text{nod}|W}^{-1}\text{Vec}\left(\widetilde{\boldsymbol{T}}_{\text{nod},s}\right)$$

$$-\frac{1}{2}\sum_{i=1}^{p_s}\ell n\det\left(\boldsymbol{\Sigma}_{W\varepsilon,i}\right) - \frac{1}{2}\sum_{i=1}^{p_s}\left(\widetilde{\boldsymbol{W}}_s^i\right)^\top \boldsymbol{\Sigma}_{W\varepsilon,i}^{-1}\widetilde{\boldsymbol{W}}_s^i . \tag{8.5.31}$$

The computational complexity of this function for repeated evaluation at different parameter settings $\boldsymbol{\Omega}^{\text{nod, all}}$ is $O(p_s^3 n^3 + p_s m^3)$. Under the assumptions $p_s \ll \min\{m_s, m_y/n\}$, (8.5.31) represents a significant computational cost savings over the unsimplified log likelihood function, which has order $O\left((m_y + m_s m)^3\right)$.

When simulator discrepancy is modeled, the log posterior distribution of $\boldsymbol{\Omega}^{\text{d, all}}$ given the standardized physical observations $\boldsymbol{Y}_{p,n}$ and the standardized simulation output $\boldsymbol{Y}_{s,n,\varepsilon}$ is calculated by summing (8.5.13) and (8.5.26):

$$\pi_{\text{d}}(\boldsymbol{\Omega}^{\text{d, all}} \,|\, \boldsymbol{Y}_{p,n}, \boldsymbol{Y}_{s,n,\varepsilon}, r_0) = p_{\text{d}}(\boldsymbol{\Omega}^{\text{d, all}}; r_0) + \ell_{\text{d}}(\boldsymbol{\Omega}^{\text{d, all}}; \widehat{\boldsymbol{Z}}_{\text{d}}, r_0)\,, \tag{8.5.32}$$

up to additive constants not depending on $\boldsymbol{\Omega}^{\text{d, all}}$.

When simulator discrepancy is assumed negligible, the log posterior distribution of $\boldsymbol{\Omega}^{\text{nod, all}}$ given $\boldsymbol{Y}_{p,n}$ and $\boldsymbol{Y}_{s,n,\varepsilon}$ is calculated by summing (8.5.14) and (8.5.31):

$$\pi_{\text{nod}}(\boldsymbol{\Omega}^{\text{nod, all}} \,|\, \boldsymbol{Y}_{p,n}, \boldsymbol{Y}_{s,n,\varepsilon}, r_0) = p_{\text{nod}}(\boldsymbol{\Omega}^{\text{nod, all}}; r_0) + \ell_{\text{nod}}(\boldsymbol{\Omega}^{\text{nod, all}}; \widehat{\boldsymbol{Z}}_{\text{nod}}, r_0)\,, \tag{8.5.33}$$

up to additive constants not depending on $\boldsymbol{\Omega}^{\text{nod, all}}$.

In *Case 1 applications* where Bayesian emulation of the simulator is desired based only on simulation output, the prior distribution and log likelihood function are modified to include only the portion of the joint statistical model pertaining to the simulation output. The parameters subject to statistical inference in this scenario are the components of $\boldsymbol{\Omega}^{s,\varepsilon} = \{\boldsymbol{\Omega}^s, \lambda_{s,\varepsilon}\}$, and $\boldsymbol{\Omega}^s$ is defined in (8.4.10). The log prior distribution (8.5.14) simplifies to

$$\begin{aligned}
p_s(\boldsymbol{\Omega}^{s,\varepsilon}) = {} & \sum_{i=1}^{p_s} \sum_{j=1}^{d} \left((a_{s,\rho}^{ij} - 1)\,\ell n(\rho_j^{x,i}) + (b_{s,\rho}^{ij} - 1)\,\ell n(1 - \rho_j^{x,i})\right) \\
& + \sum_{i=1}^{p_s} \sum_{j=d+1}^{d+q} \left((a_{s,\rho}^{ij} - 1)\,\ell n(\rho_j^{t,i}) + (b_{s,\rho}^{ij} - 1)\,\ell n(1 - \rho_j^{t,i})\right) \\
& + \sum_{i=1}^{p_s} \left((a_{s,\lambda}^{i} - 1)\,\ell n(\lambda_{s,i}) - a_{s,\lambda}^{i}\lambda_{s,i}\right) \\
& + \sum_{i=1}^{p_s} \left((a_{n,\lambda}^{i} - 1)\,\ell n(\lambda_{n,i}) - b_{n,\lambda}^{i}\lambda_{n,i}\right) \\
& + (a'_{s,\varepsilon} - 1)\,\ell n(\lambda_{s,\varepsilon}) - b'_{s,\varepsilon}\lambda_{s,\varepsilon}\,,
\end{aligned} \tag{8.5.34}$$

up to additive constants that do not depend on $\boldsymbol{\Omega}^{s,\varepsilon}$. The log likelihood function (8.5.31) reduces to

$$\ell_s\left(\boldsymbol{\Omega}^{s,\varepsilon}; \text{Vec}\left(\widetilde{\boldsymbol{W}}_s\right)\right) = -\frac{1}{2}\sum_{i=1}^{p_s} \ell n \det(\boldsymbol{\Sigma}_{W\varepsilon,i}) - \frac{1}{2}\sum_{i=1}^{p_s} \left(\widetilde{\boldsymbol{W}}_s^{i}\right)^{\top} \boldsymbol{\Sigma}_{W\varepsilon,i}^{-1} \widetilde{\boldsymbol{W}}_s^{i}\,. \tag{8.5.35}$$

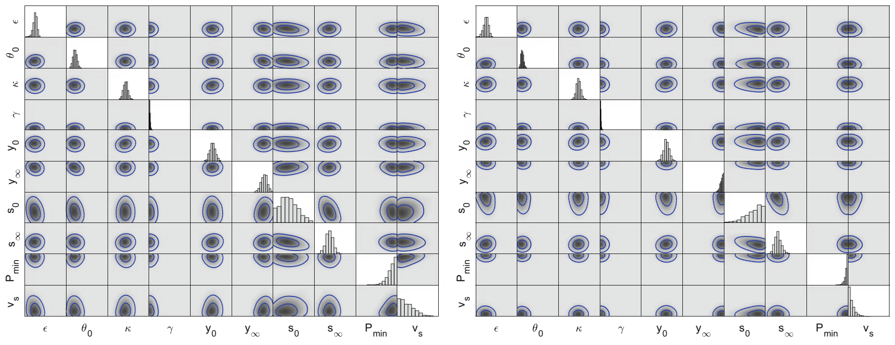


Fig. 8.20 Univariate and bivariate marginal posterior distributions for the calibration parameters $\boldsymbol{\Theta}$ under Cases 2 (left) and 3 (right)

The log posterior distribution of $\boldsymbol{\Omega}^{s,\varepsilon}$ given standardized simulation output $\boldsymbol{Y}_{s,n,\varepsilon}$ is calculated by summing (8.5.34) and (8.5.35):

$$\pi_s(\boldsymbol{\Omega}^{s,\varepsilon} \mid \boldsymbol{Y}_{s,n,\varepsilon}) = p_s(\boldsymbol{\Omega}^{s,\varepsilon}) + \ell_s\left(\boldsymbol{\Omega}^{s,\varepsilon}; \mathrm{Vec}\left(\widetilde{\boldsymbol{W}}_s\right)\right), \tag{8.5.36}$$

again up to additive constants not depending on $\boldsymbol{\Omega}^{s,\varepsilon}$.

The posterior distributions (8.5.32), (8.5.33), and (8.5.36) are not generally available in closed form. However, these distributions can be sampled using a technique such as MCMC as described in Appendix D. In particular, Metropolis within Gibbs algorithms is often effective, especially if the burn-in period is used to tune the acceptance rate of the Metropolis steps for each parameter as in Graves (2011). Such an approach may be inadequate if there is a high degree of covariance among a subset of parameters. In this case, an adaptive multivariate algorithm such as Delayed Rejection Adaptive Metropolis (Haario et al. (2006)) may be necessary.

Example 1.7 *(Continued)*. Consider *Cases 2 and 3* where emulation of the calibrated simulator is desired. The left panel of Fig. 8.20 shows univariate and bivariate *marginal posterior distributions* of the *calibration parameters* $\boldsymbol{\Theta}$ assuming model (8.4.18) (which permits simulator discrepancy); the right panel shows the analagous univariate and bivariate plots for model (8.4.20) (which assumes no discrepancy). These results are based on 1000 MCMC samples from the posterior distributions (8.5.32) and (8.5.33). The bivariate marginals were computed using kernel density estimation. As expected, some differences are present in the marginal posterior distributions between the two cases.

When the discrepancy is omitted but is not statistically zero throughout the index variable (time) domain, as in this analysis, the potential exists for the calibration process to compensate. Some parameters, particularly those exhibiting sensitivity in regions of index variable space where discrepancy is present, may be driven to incorrect solutions in an effort to minimize the unaccounted for bias that exists between the simulator and experimental data. Figure 8.21 compares the univariate marginal posterior distributions for parameters ϵ and $P_{\min}$. Free surface velocity is overall most sensitive to the equation of state perturbation ϵ, and marginally this parameter is calibrated similarly in both cases. However, the material damage parameter $P_{\min}$ is calibrated more tightly against its upper boundary in Case 3, in an attempt to minimize the unmodeled bias between the simulator and experimental data at late times (after $\approx 2.1\ \mu$s) when spall strength becomes relevant. Observed differences of this nature in calibration results for individual parameters *may* be one of several useful tools to help diagnose candidate sources of simulator inadequacy. ♦

The resulting MCMC samples can then be used to make statistical inferences of interest, such as prediction of the simulation output or the physical observations at unsampled input settings, as discussed in the next subsection.

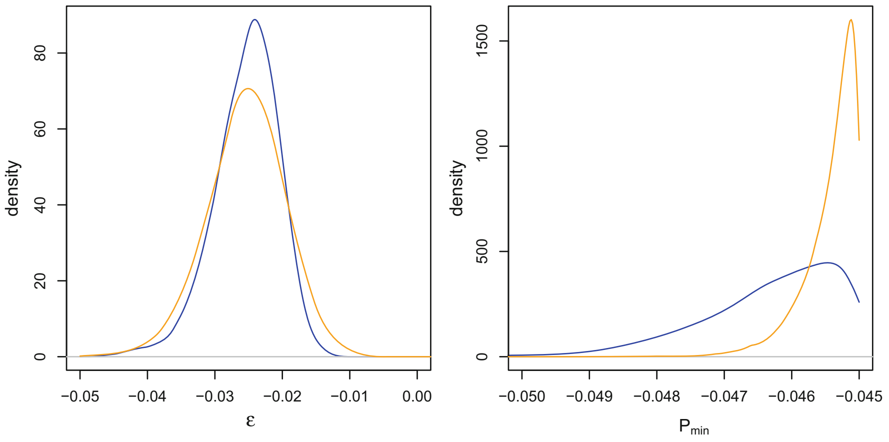


Fig. 8.21 Univariate marginal posterior distributions for ϵ (left) and $P_{\min}$ (right) under Cases 2 (blue) and 3 (orange)

8.5.2 Prediction

This subsection describes Bayesian methodology for Cases 1–3 listed at the beginning of this section. Each of these three prediction problems can be placed within a common statistical framework. Recall from Sect. 8.4.3 that, with or without simulator bias, the joint statistical model of the simulation and experimental data, which is denoted by the vector $\boldsymbol{Y}$ for this discussion, takes the form

$$\boldsymbol{Y} = \boldsymbol{C}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \tag{8.5.37}$$

where $\boldsymbol{\beta}$ and $\boldsymbol{\varepsilon}$ are mutually independent with $\boldsymbol{\beta} \sim N(\mathbf{0}, \boldsymbol{\Sigma}_\beta)$ and $\boldsymbol{\varepsilon} \sim N(\mathbf{0}, \boldsymbol{\Sigma}_\varepsilon)$. Suppose now that the unobserved quantity

$$\boldsymbol{Y}^* = \boldsymbol{C}^* \boldsymbol{\beta}^* + \boldsymbol{\varepsilon}^*, \tag{8.5.38}$$

is to be *predicted* where $\boldsymbol{\beta}^*$ and $\boldsymbol{\varepsilon}^*$ are mutually independent with $\boldsymbol{\beta}^* \sim N(\mathbf{0}, \boldsymbol{\Sigma}_{\beta^*})$ and $\boldsymbol{\varepsilon}^* \sim N(\mathbf{0}, \boldsymbol{\Sigma}_{\varepsilon^*})$. Assume further that (8.5.38) and (8.5.37) are connected via

$$\begin{pmatrix} \boldsymbol{\beta}^* \\ \boldsymbol{\beta} \end{pmatrix} \sim N\left(\mathbf{0}, \begin{bmatrix} \boldsymbol{\Sigma}_{\beta^*} & \boldsymbol{\Sigma}_{\beta^*\beta} \\ \boldsymbol{\Sigma}^\top_{\beta^*\beta} & \boldsymbol{\Sigma}_\beta \end{bmatrix}\right)$$

and that $(\boldsymbol{\beta}^*, \boldsymbol{\beta})$, $\boldsymbol{\varepsilon}^*$, and $\boldsymbol{\varepsilon}$ are mutually independent.

The joint distribution of $(\boldsymbol{Y}^*, \boldsymbol{Y})$ is readily obtained from the modeling assumptions of the previous paragraph:

$$\left[\begin{pmatrix} \boldsymbol{Y}^* \\ \boldsymbol{Y} \end{pmatrix} \middle| \mathcal{P}\right] \sim N\left(\mathbf{0}, \begin{bmatrix} \boldsymbol{\Sigma}_{\varepsilon^*} + \boldsymbol{C}^* \boldsymbol{\Sigma}_{\beta^*} (\boldsymbol{C}^*)^\top & \boldsymbol{C}^* \boldsymbol{\Sigma}_{\beta^*\beta} \boldsymbol{C}^\top \\ \boldsymbol{C} \boldsymbol{\Sigma}^\top_{\beta^*\beta} (\boldsymbol{C}^*)^\top & \boldsymbol{\Sigma}_\varepsilon + \boldsymbol{C} \boldsymbol{\Sigma}_\beta \boldsymbol{C}^\top \end{bmatrix}\right), \tag{8.5.39}$$

where $\mathcal{P}$ denotes whatever parameters in Sect. 8.4.3 are used to define $\boldsymbol{\Sigma}_\beta$ and $\boldsymbol{\Sigma}_\varepsilon$. The matrices $\boldsymbol{\Sigma}_{\beta^*}$, $\boldsymbol{\Sigma}_{\beta^*\beta}$, and $\boldsymbol{\Sigma}_{\varepsilon^*}$ are also assumed to depend only on the parameters in $\mathcal{P}$.

The *predictive distribution* of the unobserved $\boldsymbol{Y}^*$ is defined to be the conditional distribution of $\boldsymbol{Y}^*$ given $\boldsymbol{Y}$. From (8.5.39), this distribution, conditional on $\mathcal{P}$, is multivariate normal with mean vector and covariance matrix computed as described in Lemma B.2 of Appendix B:

$$[\boldsymbol{Y}^* \mid \boldsymbol{Y}, \mathcal{P}] \sim N\left(\widehat{\boldsymbol{Y}}^*, \boldsymbol{\Sigma}_{Y^*|Y}\right), \tag{8.5.40}$$

where

$$\widehat{\boldsymbol{Y}}^* = E[\boldsymbol{Y}^* | \boldsymbol{Y}, \mathcal{P}] = \left(\boldsymbol{C}^* \boldsymbol{\Sigma}_{\beta^*\beta}\right) \boldsymbol{C}^\top \left(\boldsymbol{\Sigma}_\varepsilon + \boldsymbol{C} \boldsymbol{\Sigma}_\beta \boldsymbol{C}^\top\right)^{-1} \boldsymbol{Y} \tag{8.5.41}$$

and

$$\begin{aligned} \boldsymbol{\Sigma}_{Y^*|Y} &= Cov[\boldsymbol{Y}^* | \boldsymbol{Y}, \mathcal{P}] \\ &= \boldsymbol{\Sigma}_{\varepsilon^*} + \boldsymbol{C}^* \left[\boldsymbol{\Sigma}_{\beta^*} - \boldsymbol{\Sigma}_{\beta^*\beta} \boldsymbol{C}^\top \left(\boldsymbol{\Sigma}_\varepsilon + \boldsymbol{C} \boldsymbol{\Sigma}_\beta \boldsymbol{C}^\top\right)^{-1} \boldsymbol{C} \boldsymbol{\Sigma}^\top_{\beta^*\beta}\right] (\boldsymbol{C}^*)^\top . \end{aligned} \tag{8.5.42}$$

Integrating out parameters using their posterior distribution gives the predictive *density* function of $\boldsymbol{Y}^*$ to be

$$p(\boldsymbol{Y}^* \mid \boldsymbol{Y}) = \int p(\boldsymbol{Y}^* \mid \boldsymbol{Y}, \mathcal{P})\, \pi(\mathcal{P} \mid \boldsymbol{Y})\, d\mathcal{P}, \tag{8.5.43}$$

where $p(\boldsymbol{Y}^* \mid \boldsymbol{Y}, \mathcal{P})$ is the multivariate normal density function arising from (8.5.40) and $\pi(\mathcal{P} \mid \boldsymbol{Y})$ is the posterior density of the parameters $\mathcal{P}$ conditional on the observed data $\boldsymbol{Y}$.

Often the predictive density (8.5.43) is not expressible in closed form, typically because the posterior density of $\mathcal{P}$ can be specified only up to its normalizing constant. Using MCMC to obtain samples $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_{N_{mcmc}}$ from the posterior distribution of $\mathcal{P}$, the predictive density of $\boldsymbol{Y}^*$ can be estimated as follows:

$$\hat{p}(\boldsymbol{Y}^* \mid \boldsymbol{Y}) = N_{mcmc}^{-1} \sum_{i=1}^{N_{mcmc}} p(\boldsymbol{Y}^* \mid \boldsymbol{Y}, \mathcal{P}_i)\,.$$

The estimated predictive density $\hat{p}(\boldsymbol{Y}^* \mid \boldsymbol{Y})$ is a mixture of multivariate normal density functions; it has mean

$$\overline{\boldsymbol{Y}}^*_{\hat{p}} = N_{mcmc}^{-1} \sum_{i=1}^{N_{mcmc}} \widehat{\boldsymbol{Y}}^*_i$$

for $\widehat{\boldsymbol{Y}}^*_i = E[\boldsymbol{Y}^* | \boldsymbol{Y}, \mathcal{P}_i]$, and covariance matrix

$$\boldsymbol{\Sigma}^*_{\hat{p}} = N_{mcmc}^{-1} \sum_{i=1}^{N_{mcmc}} \boldsymbol{\Sigma}^i_{Y^*|Y} + N_{mcmc}^{-1} \sum_{i=1}^{N_{mcmc}} \left(\widehat{\boldsymbol{Y}}^*_i - \overline{\boldsymbol{Y}}^*_{\hat{p}}\right)\left(\widehat{\boldsymbol{Y}}^*_i - \overline{\boldsymbol{Y}}^*_{\hat{p}}\right)^\top$$

for $\boldsymbol{\Sigma}^i_{Y^*|Y} = Cov[\boldsymbol{Y}^* | \boldsymbol{Y}, \mathcal{P}_i]$. Realizations $\boldsymbol{Y}^*_1, \boldsymbol{Y}^*_2, \ldots, \boldsymbol{Y}^*_{N_{mcmc}}$ from the predictive distribution of $\boldsymbol{Y}^*$ are obtained as follows:

$$\boldsymbol{Y}^*_i \sim N\left(\widehat{\boldsymbol{Y}}^*_i, \boldsymbol{\Sigma}^i_{Y^*|Y}\right)\,,$$

for $i = 1, \ldots, N_{mcmc}$.

The mean vector (8.5.41) and covariance matrix (8.5.42) can be simplified to take advantage of dimension reduction in prediction, analogous to the dimension reduction achieved in the log likelihood computations of Sect. 8.4.3. As before, this dimension reduction is realized by utilizing an equivalent expression for the inverse of the matrix $\boldsymbol{\Sigma}_\varepsilon + \boldsymbol{C}\boldsymbol{\Sigma}_\beta\boldsymbol{C}^\top$ obtained by substituting (8.4.34) into (8.4.32). This gives

$$\widehat{\boldsymbol{Y}}^* = \boldsymbol{C}^*\boldsymbol{\Sigma}_{\beta^*\beta}\left[\boldsymbol{\Gamma} + \left(\boldsymbol{I} - \boldsymbol{\Gamma}\boldsymbol{\Sigma}_\Gamma^{-1}\right)\left(\boldsymbol{\Sigma}_\beta + \boldsymbol{\Sigma}_\Gamma^{-1}\right)^{-1}\right]\widehat{\boldsymbol{\beta}} \tag{8.5.44}$$

$$\boldsymbol{\Sigma}_{Y^*|Y} = \boldsymbol{\Sigma}_{\varepsilon^*} + \boldsymbol{C}^*\left(\boldsymbol{\Sigma}_{\beta^*} - \boldsymbol{\Sigma}_{\beta^*\beta}\boldsymbol{\Sigma}_{\beta,\Gamma}\boldsymbol{\Sigma}^\top_{\beta^*\beta}\right)(\boldsymbol{C}^*)^\top \tag{8.5.45}$$

after some algebra, where

$$\boldsymbol{\Sigma}_{\beta,\Gamma} = \boldsymbol{\Gamma} - \boldsymbol{\Gamma}\boldsymbol{\Sigma}_\Gamma^{-1}\boldsymbol{\Gamma} + \left(\boldsymbol{I} - \boldsymbol{\Gamma}\boldsymbol{\Sigma}_\Gamma^{-1}\right)\left(\boldsymbol{\Sigma}_\beta + \boldsymbol{\Sigma}_\Gamma^{-1}\right)^{-1}\left(\boldsymbol{I} - \boldsymbol{\Sigma}_\Gamma^{-1}\boldsymbol{\Gamma}\right)$$

and the matrix $\boldsymbol{\Gamma}$ is defined below (8.4.33) and vector $\widehat{\boldsymbol{\beta}}$ by (8.4.36). In the remainder of this subsection, the approach used by Higdon et al. (2008) is adopted and the matrix $\boldsymbol{\Gamma}$ will be assumed negligible ($\boldsymbol{\Gamma} \approx \boldsymbol{0}$) *except* in the calculation of $\boldsymbol{\Sigma}_\Gamma^{-1}$. With this simplification, (8.5.44) and (8.5.45) become

$$\widehat{\boldsymbol{Y}}^* = \boldsymbol{C}^*\boldsymbol{\Sigma}_{\beta^*\beta}\left(\boldsymbol{\Sigma}_\beta + \boldsymbol{\Sigma}_\Gamma^{-1}\right)^{-1}\widehat{\boldsymbol{\beta}} \tag{8.5.46}$$

$$\boldsymbol{\Sigma}_{Y^*|Y} = \boldsymbol{\Sigma}_{\varepsilon^*} + \boldsymbol{C}^* \left(\boldsymbol{\Sigma}_{\beta^*} - \boldsymbol{\Sigma}_{\beta^*\beta} \left(\boldsymbol{\Sigma}_\beta + \boldsymbol{\Sigma}_\Gamma^{-1} \right)^{-1} \boldsymbol{\Sigma}_{\beta^*\beta}^\top \right) (\boldsymbol{C}^*)^\top . \tag{8.5.47}$$

With the above simplification, the matrix inverse calculations in (8.5.46) and (8.5.47) have order determined by the length of the reduced data vector $\widehat{\boldsymbol{\beta}}$ rather than by the length of the original data vector $\boldsymbol{Y}$, resulting in considerable computational savings.

8.5.2.1 Emulation of the Simulation Output Using Only Simulator Data

The first scenario described here is prediction of the simulation output at unsampled input sites. This prediction scenario will be restricted to the case of observing simulator data only (and physical observations are not available). In this case, $\mathcal{P} = \boldsymbol{\Omega}^{s,\varepsilon} = \{\boldsymbol{\Omega}^s, \lambda_{s,\varepsilon}\}$, and $\boldsymbol{\Omega}^s$ are defined in (8.4.10).

Suppose the simulator is to be predicted at the $\tilde{m}$ input settings $(\widetilde{\boldsymbol{x}}_1^s, \widetilde{\boldsymbol{t}}_1), (\widetilde{\boldsymbol{x}}_2^s, \widetilde{\boldsymbol{t}}_2), \ldots, (\widetilde{\boldsymbol{x}}_{\tilde{m}}^s, \widetilde{\boldsymbol{t}}_{\tilde{m}})$. Let the $\tilde{m} \times p_s$ matrix

$$\boldsymbol{W}_s^* = \left[\boldsymbol{W}^s(\widetilde{\boldsymbol{x}}_1^s, \widetilde{\boldsymbol{t}}_1) \; \boldsymbol{W}^s(\widetilde{\boldsymbol{x}}_2^s, \widetilde{\boldsymbol{t}}_2) \cdots \boldsymbol{W}^s(\widetilde{\boldsymbol{x}}_{\tilde{m}}^s, \widetilde{\boldsymbol{t}}_{\tilde{m}}) \right]^\top$$

denote the corresponding basis coefficient processes in the emulator model. To predict realizations of $\boldsymbol{W}_s^*$, consider the following settings: $\boldsymbol{C}^* = \boldsymbol{I}_{\tilde{m}p_s}$, $\boldsymbol{\beta}^* = \text{Vec}(\boldsymbol{W}_s^*)$, and $\boldsymbol{\Sigma}_{\varepsilon^*} = \boldsymbol{0}_{\tilde{m}p_s,\tilde{m}p_s}$. Therefore, $\boldsymbol{\Sigma}_{\beta^*} = \boldsymbol{\Sigma}_{W^*}$ for

$$\boldsymbol{\Sigma}_{W^*} = diag\left(\boldsymbol{\Sigma}_{W^*,1}, \boldsymbol{\Sigma}_{W^*,2}, \ldots, \boldsymbol{\Sigma}_{W^*,p_s} \right) .$$

For $k \in \{1, \ldots, p_s\}$ and $i, j = 1, \ldots, \tilde{m}$, the (i, j) element of $\boldsymbol{\Sigma}_{W^*,k}$ is

$$\frac{1}{\lambda_{s,k}} R_{s,k}\left((\widetilde{\boldsymbol{x}}_i^s, \widetilde{\boldsymbol{t}}_i), (\widetilde{\boldsymbol{x}}_j^s, \widetilde{\boldsymbol{t}}_j) \right) + \frac{1}{\lambda_{n,k}} I\left\{ (\widetilde{\boldsymbol{x}}_i^s, \widetilde{\boldsymbol{t}}_i) = (\widetilde{\boldsymbol{x}}_j^s, \widetilde{\boldsymbol{t}}_j) \right\} .$$

Because only simulator data are used for emulation, the following assignments are made: $\boldsymbol{Y} = \boldsymbol{Y}_{s,n,\varepsilon}$, $\boldsymbol{C} = \boldsymbol{X}_s$ from (8.4.29), $\boldsymbol{\beta} = \text{Vec}(\boldsymbol{W}_s)$, and $\boldsymbol{\Sigma}_\varepsilon = \lambda_{s,\varepsilon}^{-1} \boldsymbol{I}_{m_s m}$. Hence, $\boldsymbol{\Sigma}_\beta = \boldsymbol{\Sigma}_W$ from (8.4.24), and $\boldsymbol{\Sigma}_{\beta^*\beta} = \boldsymbol{\Sigma}_{W^*W}$ for

$$\boldsymbol{\Sigma}_{W^*W} = diag\left(\boldsymbol{\Sigma}_{W^*W,1}, \boldsymbol{\Sigma}_{W^*W,2}, \ldots, \boldsymbol{\Sigma}_{W^*W,p_s} \right) .$$

For $k \in \{1, \ldots, p_s\}$, $i = 1, \ldots, \tilde{m}$, and $j = 1, \ldots, m$, the (i, j) element of $\boldsymbol{\Sigma}_{W^*W,k}$ is

$$R_{s,k}\left((\widetilde{\boldsymbol{x}}_i^s, \widetilde{\boldsymbol{t}}_i), (\boldsymbol{x}_j^s, \boldsymbol{t}_j) \right) / \lambda_{s,k} .$$

Since $\boldsymbol{X}_s$ has full column rank by construction, $\boldsymbol{\Gamma} = \boldsymbol{0}_{mp_s,mp_s}$. It follows that

$$\boldsymbol{\Sigma}_\Gamma = \lambda_{s,\varepsilon} m^{-1} \left(\left(\widetilde{\boldsymbol{\Sigma}}_{11}^s \right)^2 \otimes \boldsymbol{I}_m \right) .$$

Combining (8.4.36) and (8.4.46), the reduced data vector is $\widehat{\boldsymbol{\beta}} = \text{Vec}\left(\widetilde{\boldsymbol{W}}_s\right)$. Utilizing (8.5.20), consider the $\tilde{m} \times p_s$ matrix

$$\widetilde{\boldsymbol{T}}^*_{w,s} = \left[\boldsymbol{\Sigma}_{W^*W,1}\boldsymbol{\Sigma}^{-1}_{W\varepsilon,1}\widetilde{\boldsymbol{W}}^1_s \; \boldsymbol{\Sigma}_{W^*W,2}\boldsymbol{\Sigma}^{-1}_{W\varepsilon,2}\widetilde{\boldsymbol{W}}^2_s \cdots \boldsymbol{\Sigma}_{W^*W,p_s}\boldsymbol{\Sigma}^{-1}_{W\varepsilon,p_s}\widetilde{\boldsymbol{W}}^{p_s}_s\right],$$

where $\widetilde{\boldsymbol{W}}^i_s$ is the i^{th} column of the matrix $\widetilde{\boldsymbol{W}}_s$ for $i = 1, \ldots, p_s$. Invoking (8.5.19), the mean vector (8.5.46) and covariance matrix (8.5.47) take the values

$$\begin{aligned}\widehat{\boldsymbol{Y}}^* &= \boldsymbol{\Sigma}_{W^*W}\boldsymbol{\Sigma}^{-1}_{W\varepsilon}\text{Vec}\left(\widetilde{\boldsymbol{W}}_s\right) \\ &= \text{Vec}\left(\widetilde{\boldsymbol{T}}^*_{w,s}\right) \end{aligned} \tag{8.5.48}$$

$$\begin{aligned}\boldsymbol{\Sigma}_{Y^*|Y} &= \boldsymbol{\Sigma}_{W^*} - \boldsymbol{\Sigma}_{W^*W}\boldsymbol{\Sigma}^{-1}_{W\varepsilon}\boldsymbol{\Sigma}^{\top}_{W^*W} \\ &= diag\left(\boldsymbol{\Sigma}_{W^*,i} - \boldsymbol{\Sigma}_{W^*W,i}\boldsymbol{\Sigma}^{-1}_{W\varepsilon,i}\boldsymbol{\Sigma}^{\top}_{W^*W,i},\; i = 1, \ldots, p_s\right).\end{aligned} \tag{8.5.49}$$

For given parameter settings $\mathcal{P}$, the elements of $\widehat{\boldsymbol{Y}}^*$ and $\boldsymbol{\Sigma}_{Y^*|Y}$ corresponding to each test site take the form of the BLUP (3.2.7) and its variance (3.2.8) when the regression component is zeroed out, calculated individually for each basis coefficient.

Suppose N_{mcmc} draws $\boldsymbol{\Omega}^{s,\varepsilon}_1, \ldots, \boldsymbol{\Omega}^{s,\varepsilon}_{N_{mcmc}}$ have been sampled from the posterior distribution (8.5.36). The i^{th} sample, $i = 1, \ldots, N_{mcmc}$, from the basis coefficient processes is obtained by generating

$$\boldsymbol{Y}^*_{i,j} \sim N_{\tilde{m}}\left(\boldsymbol{\Sigma}_{W^*W,j}\boldsymbol{\Sigma}^{-1}_{W\varepsilon,j}\widetilde{\boldsymbol{W}}^j_s,\; \boldsymbol{\Sigma}_{W^*,j} - \boldsymbol{\Sigma}_{W^*W,j}\boldsymbol{\Sigma}^{-1}_{W\varepsilon,j}\boldsymbol{\Sigma}^{\top}_{W^*W,j}\right) \tag{8.5.50}$$

for $j = 1, \ldots, p_s$, using the draw $\boldsymbol{\Omega}^{s,\varepsilon}_i$ to calculate the mean vector and covariance matrix of the multivariate normal distribution in (8.5.50). Sampling proceeds through $i = 1, \ldots, N_{mcmc}$. Each of the N_{mcmc} posterior realizations of $\boldsymbol{\Omega}^{s,\varepsilon}$ requires $O((m^3 + \tilde{m}^3)p_s)$ operations to generate a sample $\left\{\boldsymbol{Y}^*_{i,j}\right\}^{p_s}_{j=1}$, which is only linear in the number of simulator basis vectors retained due to the block diagonal structure of the covariance matrix (8.5.49).

The samples of the basis coefficients are then easily transformed into samples from the emulator itself. For the i^{th} posterior realization $\boldsymbol{\Omega}^{s,\varepsilon}_i$, arrange the p_s basis coefficient samples (8.5.50) into the $p_s \times \tilde{m}$ matrix $\mathcal{W}^*_i$,

$$\mathcal{W}^*_i = \left[\boldsymbol{Y}^*_{i,1}\; \boldsymbol{Y}^*_{i,2} \ldots \boldsymbol{Y}^*_{i,p_s}\right]^{\top}.$$

Using $\boldsymbol{K}^s$ defined in (8.4.4), the $\tilde{m}$ columns of the matrix

$$\boldsymbol{Y}^{*,i}_{s,n} = \boldsymbol{K}^s\mathcal{W}^*_i$$

are samples from the emulator on the standardized scale, corresponding to the input settings $(\widetilde{\boldsymbol{x}}^s_1, \widetilde{\boldsymbol{t}}_1), (\widetilde{\boldsymbol{x}}^s_2, \widetilde{\boldsymbol{t}}_2), \ldots, (\widetilde{\boldsymbol{x}}^s_{\tilde{m}}, \widetilde{\boldsymbol{t}}_{\tilde{m}})$. In the functional case, these samples are transformed back to the original scale using (8.4.1) and (8.4.2) as follows

$$\boldsymbol{Y}_s^{*,i} = \varsigma^s \boldsymbol{Y}_{s,n}^{*,i} + \overline{\boldsymbol{y}}^s \mathbf{1}_{\tilde{m}}^\top , \tag{8.5.51}$$

while this transformation is modified slightly in the multivariate case using (8.4.3) in place of (8.4.2),

$$\boldsymbol{Y}_s^{*,i} = \left(\boldsymbol{C}_d^s\right)^{1/2} \boldsymbol{Y}_{s,n}^{*,i} + \overline{\boldsymbol{y}}^s \mathbf{1}_{\tilde{m}}^\top .$$

Upon completion of the sampling process, N_{mcmc} emulator samples will have been generated for each of the $\tilde{m}$ input settings.

Example 1.7 *(Continued)*. To illustrate Case 1 emulation, Fig. 8.22 shows the centered training and test data, along with $N_{mcmc} = 1000$ realizations of the centered emulator evaluated at the parameter setting that generated the test set. Using (8.5.51) and noting that $\tilde{m} = 1$, the i^{th} realization of the centered emulator is given by the vector $\boldsymbol{Y}_s^{*,i} - \overline{\boldsymbol{y}}^s$.

The slight overprediction in the flat portion of the velocity profile seen in Fig. 8.16 is significant relative to uncertainty in the emulator, while the velocity behavior in the more variable regions of the time domain is captured adequately. This suggests that *more than* three simulator basis vectors would be required to accurately capture the low-variability constant velocity portion of the time domain for this test run and presumably other test runs. ♦

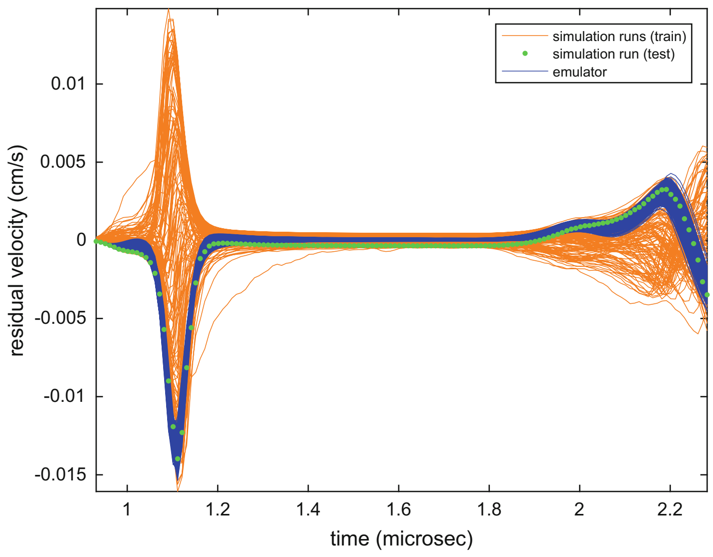


Fig. 8.22 Centered simulation training runs (orange lines), centered simulation test run (green dots), and centered emulator realizations (blue lines)

If desired, simulator basis noise can be added to $\boldsymbol{Y}_{s,n}^{*,i}$ before transformation back to the original scale. Generate independent samples

$$\boldsymbol{\varepsilon}_j^{*,i} \sim N_{m_s}\left(\mathbf{0}_{m_s}, \frac{1}{\lambda_{s,\varepsilon}^i}\boldsymbol{I}_{m_s}\right)$$

for $j = 1, \ldots, \tilde{m}$, where m_s is the size of the input variable(s) mesh and $\lambda_{s,\varepsilon}^i$ is the realized value of $\lambda_{s,\varepsilon}$ from $\boldsymbol{\Omega}_i^{s,\varepsilon}$. Collect these samples into the $m_s \times \tilde{m}$ matrix $\mathcal{E}_i^*$,

$$\mathcal{E}_i^* = \left[\boldsymbol{\varepsilon}_1^{*,i}\ \boldsymbol{\varepsilon}_2^{*,i} \cdots \boldsymbol{\varepsilon}_{\tilde{m}}^{*,i}\right],$$

calculate

$$\boldsymbol{Y}_{s,n,\varepsilon}^{*,i} = \boldsymbol{Y}_{s,n}^{*,i} + \mathcal{E}_i^*,$$

and substitute $\boldsymbol{Y}_{s,n,\varepsilon}^{*,i}$ for $\boldsymbol{Y}_{s,n}^{*,i}$ in the appropriate transformation of the previous paragraph.

8.5.2.2 Emulation of the Calibrated Simulator Output Modeling the Simulator Bias

The second scenario considered here is prediction of the calibrated simulator output at unsampled input sites when simulator bias is accounted for statistically, as described in Sect. 8.4.2. This prediction application uses both simulation output and physical observations. In this case, $\mathcal{P} = \boldsymbol{\Omega}^{\text{d, all}} = \{\boldsymbol{\Omega}^{p,\text{d},\varepsilon}, \lambda_{s,\varepsilon}\}$ and $\boldsymbol{\Omega}^{p,\text{d},\varepsilon}$ is defined in (8.4.19).

Suppose realizations of the calibrated simulator are desired at the $\tilde{n}$ input settings $\widetilde{\boldsymbol{x}}_1^p, \widetilde{\boldsymbol{x}}_2^p, \ldots, \widetilde{\boldsymbol{x}}_{\tilde{n}}^p$. The corresponding basis coefficient processes in the calibrated simulator model are collected in the $\tilde{n} \times p_s$ matrix $\boldsymbol{U}_s^*$:

$$\boldsymbol{U}_s^* = \left[\boldsymbol{W}^s(\widetilde{\boldsymbol{x}}_1^p, \boldsymbol{\Theta})\ \boldsymbol{W}^s(\widetilde{\boldsymbol{x}}_2^p, \boldsymbol{\Theta}) \cdots \boldsymbol{W}^s(\widetilde{\boldsymbol{x}}_{\tilde{n}}^p, \boldsymbol{\Theta})\right]^\top. \tag{8.5.52}$$

Notice that the random calibration parameter $\boldsymbol{\Theta}$ has replaced the user-specified calibration parameter settings in the calibrated simulator. The discrepancy basis coefficient processes associated with the above input settings are collected in the $\tilde{n} \times p_\delta$ matrix

$$\boldsymbol{V}_p^* = \left[\boldsymbol{V}^p(\widetilde{\boldsymbol{x}}_1^p)\ \boldsymbol{V}^p(\widetilde{\boldsymbol{x}}_2^p) \cdots \boldsymbol{V}^p(\widetilde{\boldsymbol{x}}_{\tilde{n}}^p)\right]^\top.$$

To predict realizations of these matrices, set $\boldsymbol{C}^* = \boldsymbol{I}_{\tilde{n}(p_\delta+p_s)}$, $\boldsymbol{\beta}^* = \text{Vec}\left(\left[\boldsymbol{V}_p^*\ \boldsymbol{U}_s^*\right]\right)$, and $\boldsymbol{\Sigma}_{\varepsilon^*} = \mathbf{0}_{\tilde{n}(p_\delta+p_s),\tilde{n}(p_\delta+p_s)}$. By the definition of $\boldsymbol{\beta}^*$, $\boldsymbol{\Sigma}_{\beta^*} = diag\,(\boldsymbol{\Sigma}_{V^*}, \boldsymbol{\Sigma}_{U^*})$ for

$$\begin{aligned}\boldsymbol{\Sigma}_{V^*} &= diag\left(\boldsymbol{\Sigma}_{V^*,1}, \boldsymbol{\Sigma}_{V^*,2}, \ldots, \boldsymbol{\Sigma}_{V^*,p_\delta}\right) \text{ and} \\ \boldsymbol{\Sigma}_{U^*} &= diag\left(\boldsymbol{\Sigma}_{U^*,1}, \boldsymbol{\Sigma}_{U^*,2}, \ldots, \boldsymbol{\Sigma}_{U^*,p_s}\right).\end{aligned} \tag{8.5.53}$$

For $k \in \{1, \ldots, p_\delta\}$ and $i, j = 1, \ldots, \tilde{n}$, the (i, j) element of $\boldsymbol{\Sigma}_{V^*,k}$, $i, j = 1, \ldots, \tilde{n}$ is

$$R_{\delta,l}(\widetilde{\boldsymbol{x}}_i^p, \widetilde{\boldsymbol{x}}_j^p)/\lambda_{\delta,l},$$

where $l \in \{1, \ldots, F\}$ is chosen so that $k \in G_l$. For $k \in \{1, \ldots, p_s\}$ and $i, j = 1, \ldots, \tilde{n}$, the (i, j) element of $\boldsymbol{\Sigma}_{U^*,k}$ is

$$\frac{1}{\lambda_{s,k}} R_{s,k}\left((\widetilde{\boldsymbol{x}}_i^p, \boldsymbol{\Theta}), (\widetilde{\boldsymbol{x}}_j^p, \boldsymbol{\Theta})\right) + \frac{1}{\lambda_{n,k}} I\{\widetilde{\boldsymbol{x}}_i^p = \widetilde{\boldsymbol{x}}_j^p\}.$$

Because both simulation output and physical observations are used in this case, $\boldsymbol{Y} = \left(\boldsymbol{Y}_{p,n}^\top, \boldsymbol{Y}_{s,n,\varepsilon}^\top\right)^\top$ and $\boldsymbol{\Sigma}_\varepsilon = \boldsymbol{\Sigma}_E$ from (8.4.27). Combining (8.4.22), (8.4.23), (8.4.28), and (8.4.29) gives

$$\boldsymbol{C} = \begin{pmatrix} \boldsymbol{X}_{p,\mathrm{d}} & \boldsymbol{0}_{m_y, mp_s} \\ \boldsymbol{0}_{m_s m, n(p_\delta + p_s)} & \boldsymbol{X}_s \end{pmatrix},$$

$\boldsymbol{\beta} = \boldsymbol{Z}_\mathrm{d}$, $\boldsymbol{\Sigma}_\beta = \boldsymbol{\Sigma}_{Z,\mathrm{d}}$, and $\boldsymbol{\Sigma}_{\beta^*\beta} = \boldsymbol{\Sigma}_{Z^*Z}$ for

$$\boldsymbol{\Sigma}_{Z^*Z} = \begin{bmatrix} \boldsymbol{\Sigma}_{V^*V} & \boldsymbol{0}_{\tilde{n}p_\delta, np_s} & \boldsymbol{0}_{\tilde{n}p_\delta, mp_s} \\ \boldsymbol{0}_{\tilde{n}p_s, np_\delta} & \boldsymbol{\Sigma}_{U^*U} & \boldsymbol{\Sigma}_{U^*W} \end{bmatrix}, \tag{8.5.54}$$

with

$$\boldsymbol{\Sigma}_{V^*V} = diag\left(\boldsymbol{\Sigma}_{V^*V,1}, \boldsymbol{\Sigma}_{V^*V,2}, \ldots, \boldsymbol{\Sigma}_{V^*V,p_\delta}\right),$$

$$\boldsymbol{\Sigma}_{U^*U} = diag\left(\boldsymbol{\Sigma}_{U^*U,1}, \boldsymbol{\Sigma}_{U^*U,2}, \ldots, \boldsymbol{\Sigma}_{U^*U,p_s}\right), \tag{8.5.55}$$

$$\boldsymbol{\Sigma}_{U^*W} = diag\left(\boldsymbol{\Sigma}_{U^*W,1}, \boldsymbol{\Sigma}_{U^*W,2}, \ldots, \boldsymbol{\Sigma}_{U^*W,p_s}\right). \tag{8.5.56}$$

For $k \in \{1, \ldots, p_\delta\}$, $i = 1, \ldots, \tilde{n}$, and $j = 1, \ldots, n$, the (i, j) element of $\boldsymbol{\Sigma}_{V^*V,k}$ is

$$R_{\delta,l}(\widetilde{\boldsymbol{x}}_i^p, \boldsymbol{x}_j^p)/\lambda_{\delta,l},$$

where $l \in \{1, \ldots, F\}$ is chosen so that $k \in G_l$. For $k \in \{1, \ldots, p_s\}$, $i = 1, \ldots, \tilde{n}$, and $j = 1, \ldots, n$, the (i, j) element of $\boldsymbol{\Sigma}_{U^*U,k}$ is

$$R_{s,k}\left((\widetilde{\boldsymbol{x}}_i^p, \boldsymbol{\Theta}), (\boldsymbol{x}_j^p, \boldsymbol{\Theta})\right)/\lambda_{s,k}.$$

For $k \in \{1, \ldots, p_s\}$, $i = 1, \ldots, \tilde{n}$, and $j = 1, \ldots, m$, the (i, j) element of $\boldsymbol{\Sigma}_{U^*W,k}$ is

$$R_{s,k}\left((\widetilde{\boldsymbol{x}}_i^p, \boldsymbol{\Theta}), (\boldsymbol{x}_j^s, \boldsymbol{t}_j)\right)/\lambda_{s,k}.$$

Supposing, as discussed in Sect. 8.4.3, that the column space of $\boldsymbol{X}_{p,\mathrm{d}}$ need not be of full rank. The form of $\boldsymbol{\Gamma}$ adopted in (8.4.37) is also used in this prediction scenario, namely,

$$\boldsymbol{\Gamma} = diag\left(\lambda_{p,\varepsilon} r_0 \boldsymbol{I}_{n(p_\delta + p_s)}, \boldsymbol{0}_{mp_s, mp_s}\right).$$

It follows that $\boldsymbol{\Sigma}_\Gamma = \boldsymbol{\Sigma}_{\Gamma,\mathrm{d}}$ from (8.4.39).

Applying the notation in (8.4.36) and (8.4.48), the reduced data vector $\widehat{\boldsymbol{\beta}} = \widehat{\boldsymbol{Z}}_\mathrm{d}$. Let

$$\boldsymbol{\Sigma}_{Z^*VU|W} = diag\left(\boldsymbol{\Sigma}_{V^*V}, \boldsymbol{\Sigma}_{U^*U|W}\right) \tag{8.5.57}$$

where

$$\boldsymbol{\Sigma}_{U^*U|W} = \boldsymbol{\Sigma}_{U^*U} - \boldsymbol{\Sigma}_{U^*W}\boldsymbol{\Sigma}_{W\varepsilon}^{-1}\boldsymbol{\Sigma}_{UW}^{\top}\,, \tag{8.5.58}$$

using (8.4.25) and (8.5.19). Invoking (8.5.22) and (8.5.23), the mean vector (8.5.46) is given by

$$\widehat{\boldsymbol{Y}}^* = \boldsymbol{\Sigma}_{Z^*VU|W}\boldsymbol{\Sigma}_{\mathrm{d}|W}^{-1}\mathrm{Vec}\left(\left[\widetilde{\boldsymbol{V}}_p\ \widetilde{\boldsymbol{T}}_{\mathrm{d},s}\right]\right) + \mathrm{Vec}\left(\left[\boldsymbol{0}_{\tilde{n},p_\delta}\ \widetilde{\boldsymbol{T}}^*_{u,s}\right]\right)\,, \tag{8.5.59}$$

where the $\tilde{n} \times p_s$ matrix $\widetilde{\boldsymbol{T}}^*_{u,s}$ is obtained using (8.5.20)

$$\widetilde{\boldsymbol{T}}^*_{u,s} = \left[\boldsymbol{\Sigma}_{U^*W,1}\boldsymbol{\Sigma}_{W\varepsilon,1}^{-1}\widetilde{\boldsymbol{W}}_s^1\ \boldsymbol{\Sigma}_{U^*W,2}\boldsymbol{\Sigma}_{W\varepsilon,2}^{-1}\widetilde{\boldsymbol{W}}_s^2\ \cdots\ \boldsymbol{\Sigma}_{U^*W,p_s}\boldsymbol{\Sigma}_{W\varepsilon,p_s}^{-1}\widetilde{\boldsymbol{W}}_s^{p_s}\right]\,, \tag{8.5.60}$$

and $\widetilde{\boldsymbol{W}}_s^i$ is the i^{th} column of the matrix $\widetilde{\boldsymbol{W}}_s$ for $i = 1, \ldots, p_s$. The second term on the right-hand side of $\widehat{\boldsymbol{Y}}^*$ predicts the calibrated simulator using only simulation output. The first term adjusts these predictions by bringing in additional information simultaneously about discrepancy and the calibrated simulator directly from the experimental data.

Consider $\boldsymbol{\Sigma}_{Z^*|W} = diag\left(\boldsymbol{\Sigma}_{V^*}, \boldsymbol{\Sigma}_{U^*|W}\right)$ where

$$\boldsymbol{\Sigma}_{U^*|W} = \boldsymbol{\Sigma}_{U^*} - \boldsymbol{\Sigma}_{U^*W}\boldsymbol{\Sigma}_{W\varepsilon}^{-1}\boldsymbol{\Sigma}_{U^*W}^{\top}\,. \tag{8.5.61}$$

The covariance matrix (8.5.47) is given by

$$\boldsymbol{\Sigma}_{Y^*|Y} = \boldsymbol{\Sigma}_{Z^*|W} - \boldsymbol{\Sigma}_{Z^*VU|W}\boldsymbol{\Sigma}_{\mathrm{d}|W}^{-1}\boldsymbol{\Sigma}_{Z^*VU|W}^{\top}\,. \tag{8.5.62}$$

The first term in $\boldsymbol{\Sigma}_{Y^*|Y}$ represents uncertainty in discrepancy and calibrated simulator prediction using only simulation output, which is reduced by incorporating information from the experimental data in the second term.

Suppose N_{mcmc} draws $\boldsymbol{\Omega}_1^{\mathrm{d,\,all}}, \ldots, \boldsymbol{\Omega}_{N_{mcmc}}^{\mathrm{d,\,all}}$ have been sampled from the posterior distribution (8.5.32). The i^{th} sample from the basis coefficient processes is obtained by generating

$$\boldsymbol{Y}_i^* \sim N_{\tilde{n}(p_\delta+p_s)}\left(\widehat{\boldsymbol{Y}}^*, \boldsymbol{\Sigma}_{Y^*|Y}\right)\,,$$

using the realized values $\boldsymbol{\Omega}_i^{\mathrm{d,\,all}}$ to calculate the mean vector (8.5.59) and covariance matrix (8.5.62) of this multivariate normal distribution. Sampling proceeds through $i = 1, \ldots, N_{mcmc}$. The $\tilde{n}(p_\delta + p_s) \times 1$ vector $\boldsymbol{Y}_i^*$ can be written

$$\boldsymbol{Y}_i^* = \mathrm{Vec}\left(\left[\left(\mathcal{V}_i^*\right)^{\top}\ \left(\mathcal{U}_i^*\right)^{\top}\right]\right)\,,$$

where

$$\mathcal{V}_i^* = \left[\boldsymbol{Y}_{i,1}^*\ \boldsymbol{Y}_{i,2}^*\ \cdots\ \boldsymbol{Y}_{i,p_\delta}^*\right]^{\top} \text{ and}$$

$$\mathcal{U}_i^* = \left[\boldsymbol{Y}_{i,p_\delta+1}^*\ \boldsymbol{Y}_{i,p_\delta+2}^*\ \cdots\ \boldsymbol{Y}_{i,p_\delta+p_s}^*\right]^{\top}$$

for $\tilde{n} \times 1$ vectors $\boldsymbol{Y}_{i,j}^*$, $j = 1, \ldots, p_\delta + p_s$.

Using (8.4.4), the $\tilde{n}$ columns of the matrix

$$\boldsymbol{M}_{u,n}^{*,i} = \boldsymbol{K}^s \boldsymbol{\mathcal{U}}_i^*$$

are samples from the calibrated emulator on the standardized scale, and

$$\boldsymbol{M}_{v,n}^{*,i} = \boldsymbol{D}^p \boldsymbol{\mathcal{V}}_i^*$$

are samples from the discrepancy process on the standardized scale, corresponding to the input settings $\widetilde{\boldsymbol{x}}_1^p, \widetilde{\boldsymbol{x}}_2^p, \ldots, \widetilde{\boldsymbol{x}}_{\tilde{n}}^p$. The input variable(s) mesh associated with the rows of $\boldsymbol{K}^s$ is also utilized in the formulation of $\boldsymbol{D}^p$, as discussed in Sect. 8.4.2 with regard to distinctions between the functional and multivariate data settings. In the functional case, these samples are transformed back to the original scale using (8.4.1) and (8.4.2) as follows:

$$\boldsymbol{M}_u^{*,i} = \varsigma^s \boldsymbol{M}_{u,n}^{*,i} + \overline{\boldsymbol{y}}^s \boldsymbol{1}_{\tilde{n}}^\top , \tag{8.5.63}$$

$$\boldsymbol{M}_v^{*,i} = \varsigma^s \boldsymbol{M}_{v,n}^{*,i} , \tag{8.5.64}$$

while this transformation is modified slightly in the multivariate case using (8.4.3) in place of (8.4.2)

$$\boldsymbol{M}_u^{*,i} = \left(\boldsymbol{C}_d^s\right)^{1/2} \boldsymbol{M}_{u,n}^{*,i} + \overline{\boldsymbol{y}}^s \boldsymbol{1}_{\tilde{n}}^\top ,$$

$$\boldsymbol{M}_v^{*,i} = \left(\boldsymbol{C}_d^s\right)^{1/2} \boldsymbol{M}_{v,n}^{*,i} .$$

Upon completion of the sampling process, N_{mcmc} samples from the calibrated emulator and discrepancy process will have been generated for each of the $\tilde{n}$ input settings.

Example 1.7 *(Continued).* For this Case 2 application, Fig. 8.23 shows the centered data for the $n = 1$ experiment superimposed on the $m = 128$ centered simulation runs and $N_{mcmc} = 1000$ realizations of the centered, calibrated emulator. Using (8.5.63) and noting that $\tilde{n} = 1$, the i^{th} realization of the centered, calibrated emulator is given by the vector $\boldsymbol{M}_u^{*,i} - \overline{\boldsymbol{y}}^s$.

Figure 8.24 shows 1000 realizations of the discrepancy. Using (8.5.64), the i^{th} realization of the discrepancy is given by the vector $\boldsymbol{M}_v^{*,i}$. Simulator bias is clearly present at early and late times, as indicated by systematic deviations of the discrepancy realizations from the zero line. ♦

The predicted values of expected experimental output for the i^{th} sample at the $\tilde{n}$ input settings are given by

$$\boldsymbol{M}_p^{*,i} = \boldsymbol{M}_u^{*,i} + \boldsymbol{M}_v^{*,i} . \tag{8.5.65}$$

Example 1.7 *(Continued).* Continuing the Case 2 illustration, Fig. 8.25 shows centered data for $n = 1$ experiment, superimposed on the $m = 128$ centered simulation runs, and $N_{mcmc} = 1000$ realizations of centered expected experimental output. Using (8.5.65) and noting that $\tilde{n} = 1$, the i^{th} realization of centered expected experimental output is given by the vector $\boldsymbol{M}_p^{*,i} - \overline{\boldsymbol{y}}^s$.

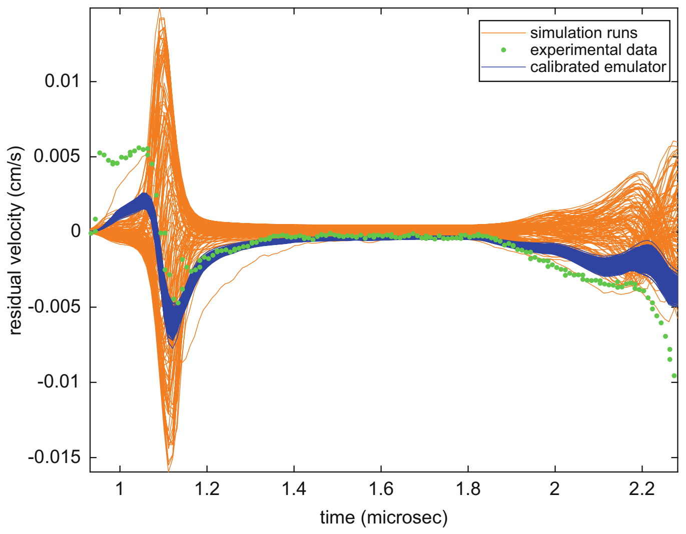


Fig. 8.23 Centered simulation runs (orange lines), centered experimental data (green dots), and centered, calibrated emulator realizations (blue lines)

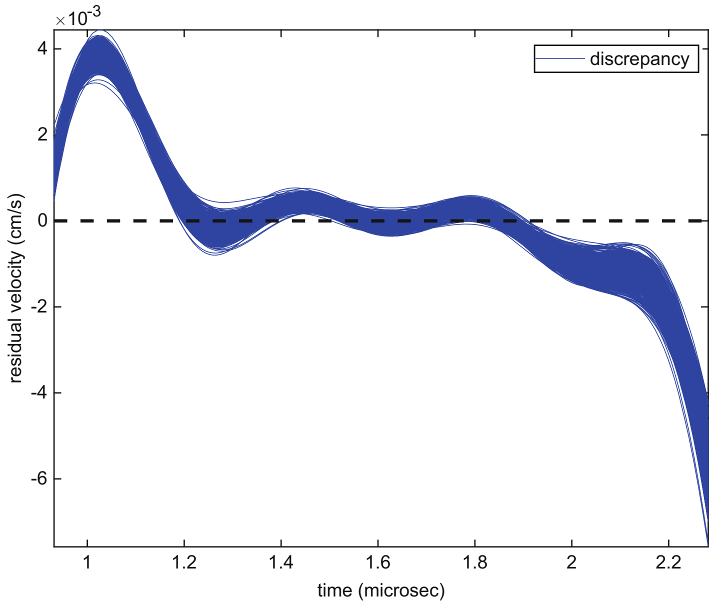


Fig. 8.24 Centered discrepancy realizations (blue lines)

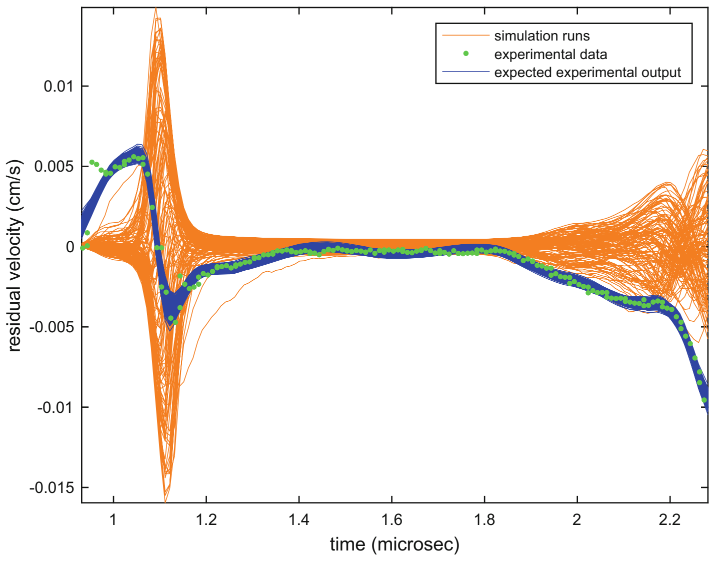


Fig. 8.25 Centered simulation runs (orange lines), centered experimental data (green dots), and centered expected experimental output realizations (blue lines)

Therefore, the discrepancy adjustment of the calibrated emulator appears to provide an adequate fit to expected experimental output except at very early times, where the abrupt velocity jump in the experimental data is modeled at low fidelity by the chosen discrepancy basis representation. This suggests that the posterior distribution of the calibration parameters $\boldsymbol{\Theta}$ may be used for uncertainty quantification in other applications involving these parameters, assuming it has been validated for predicting independently generated test data. ♦

Predicted values of experimental output itself are obtained by adding observation error to the expected experimental output (8.5.65). This is accomplished by assuming observation errors pertaining to future experiments can be independently generated from a process having characteristics similar to that inferred from past experimental output. To this end, assume

$$\boldsymbol{\varepsilon}^p(\widetilde{\boldsymbol{x}}_j^p) \sim N_{m_s}\left(\mathbf{0}_{m_s}, \frac{1}{\lambda_{p,\varepsilon}}\left(\widetilde{\boldsymbol{P}}_j^p\right)^{-1}\right),\ j = 1, \ldots, \tilde{n}, \tag{8.5.66}$$

where m_s is the size of the input variable(s) mesh and $\widetilde{\boldsymbol{P}}_j^p$ is an $m_s \times m_s$ fixed precision matrix influencing the size of observation errors at the j^{th} input setting $\widetilde{\boldsymbol{x}}_j^p$. Let the j^{th} column of the $m_s \times \tilde{n}$ error matrix $\mathcal{E}_i^*$ be a sample from the $\boldsymbol{\varepsilon}^p(\widetilde{\boldsymbol{x}}_j^p)$ process for $j = 1, \ldots, \tilde{n}$, assuming $\lambda_{p,\varepsilon}$ takes its realized value $\lambda_{p,\varepsilon}^i$ from $\boldsymbol{\Omega}_i^{\text{d, all}}$. Predicted experimental output for the i^{th} sample is then given by

$$\boldsymbol{Y}_p^{*,i} = \boldsymbol{M}_p^{*,i} + \mathcal{E}_i^*\,.$$

8.5.2.3 Emulation of the Calibrated Simulation Output Assuming No Simulator Bias

The final prediction scenario is to predict calibrated simulator output at unsampled input sites when simulator bias is assumed negligible and is omitted from the statistical modeling framework. As in the second scenario, realizations of the calibrated simulator are desired at the $\tilde{n}$ input settings $\widetilde{\boldsymbol{x}}_1^p, \widetilde{\boldsymbol{x}}_2^p, \ldots, \widetilde{\boldsymbol{x}}_{\tilde{n}}^p$. Also this prediction application uses both simulation output and physical observations. In this case, $\mathcal{P} = \boldsymbol{\Omega}^{\text{nod, all}} = \{\boldsymbol{\Omega}^{p,\text{nod},\varepsilon}, \lambda_{s,\varepsilon}\}$ and $\boldsymbol{\Omega}^{p,\text{nod},\varepsilon}$ is defined in (8.4.21). This scenario closely follows the development of the second scenario, and thus notation previously introduced in that context is utilized below as required.

To predict realizations of $\boldsymbol{U}_s^*$ from (8.5.52), consider the following settings: $\boldsymbol{C}^* = \boldsymbol{I}_{\tilde{n}p_s}$, $\boldsymbol{\beta}^* = \text{Vec}\,(\boldsymbol{U}_s^*)$, and $\boldsymbol{\Sigma}_{\varepsilon^*} = \boldsymbol{0}_{\tilde{n}p_s,\tilde{n}p_s}$. Hence, $\boldsymbol{\Sigma}_{\beta^*} = \boldsymbol{\Sigma}_{U^*}$ from (8.5.53). Because both simulation output and physical observations are available, $\boldsymbol{Y} = \left(\boldsymbol{Y}_{p,n}^\top, \boldsymbol{Y}_{s,n,\varepsilon}^\top\right)^\top$, and $\boldsymbol{\Sigma}_\varepsilon = \boldsymbol{\Sigma}_E$ from (8.4.27). Recalling (8.4.29), (8.4.52), (8.4.53), (8.4.55), (8.5.55), and (8.5.56) gives

$$\boldsymbol{C} = \begin{pmatrix} \boldsymbol{X}_{p,\text{nod}} & \boldsymbol{0}_{m_y,mp_s} \\ \boldsymbol{0}_{m_sm,np_s} & \boldsymbol{X}_s \end{pmatrix},$$

$\boldsymbol{\beta} = \boldsymbol{Z}_{\text{nod}}$, $\boldsymbol{\Sigma}_\beta = \boldsymbol{\Sigma}_{Z,\text{nod}}$, and $\boldsymbol{\Sigma}_{\beta^*\beta} = \boldsymbol{\Sigma}_{U^*Z} = \left[\boldsymbol{\Sigma}_{U^*U}\ \boldsymbol{\Sigma}_{U^*W}\right]$. The column space of $\boldsymbol{X}_{p,\text{nod}}$ will generally be of full rank. To allow for the rare possibility that this is not the case, the form of $\boldsymbol{\Gamma}$ adopted in (8.4.56) is also used in this prediction scenario, namely, $\boldsymbol{\Gamma} = diag\left(\lambda_{p,\varepsilon} r_0 \boldsymbol{I}_{np_s}, \boldsymbol{0}_{mp_s,mp_s}\right)$. It follows that $\boldsymbol{\Sigma}_\Gamma = \boldsymbol{\Sigma}_{\Gamma,\text{nod}}$ from (8.4.58). Typical applications of this prediction scenario will allow $r_0 = 0$.

Applying (8.4.36) and (8.4.64), the reduced data vector is $\widehat{\boldsymbol{\beta}} = \widehat{\boldsymbol{Z}}_{\text{nod}}$. Further, invoking (8.5.27), (8.5.28), (8.5.58), and (8.5.60), the mean vector (8.5.46) is given by

$$\widehat{\boldsymbol{Y}}^* = \boldsymbol{\Sigma}_{U^*U|W}\boldsymbol{\Sigma}_{\text{nod}|W}^{-1}\text{Vec}\left(\widetilde{\boldsymbol{T}}_{\text{nod},s}\right) + \text{Vec}\left(\widetilde{\boldsymbol{T}}_{u,s}^*\right). \tag{8.5.67}$$

The second term on the right-hand side of $\widehat{\boldsymbol{Y}}^*$ predicts the calibrated simulator using only simulation output. The first term adjusts these predictions by bringing in additional information about the calibrated simulator directly from the experimental data.

Using (8.5.61) the covariance matrix in (8.5.47) is given by

$$\boldsymbol{\Sigma}_{Y^*|Y} = \boldsymbol{\Sigma}_{U^*|W} - \boldsymbol{\Sigma}_{U^*U|W}\boldsymbol{\Sigma}_{\text{nod}|W}^{-1}\boldsymbol{\Sigma}_{U^*U|W}^\top. \tag{8.5.68}$$

The first term in $\boldsymbol{\Sigma}_{Y^*|Y}$ represents uncertainty in calibrated simulator prediction using only simulation output, which is reduced by incorporating information from the experimental data in the second term.

Suppose that N_{mcmc} draws $\boldsymbol{\Omega}_1^{\text{nod, all}}, \ldots, \boldsymbol{\Omega}_{N_{mcmc}}^{\text{nod, all}}$ have been sampled from the posterior distribution (8.5.33). Then the i^{th} sample from the basis coefficient processes is obtained by generating

$$\boldsymbol{Y}_i^* \sim N_{\tilde{n}p_s}\left(\widehat{\boldsymbol{Y}}^*, \boldsymbol{\Sigma}_{Y^*|Y}\right),$$

using the realized values $\boldsymbol{\Omega}_i^{\text{nod, all}}$ to calculate the mean vector (8.5.67) and covariance matrix (8.5.68) of this multivariate normal distribution. Proceed sampling in the order $i = 1, \ldots, N_{mcmc}$. The $\tilde{n}p_s \times 1$ vector $\boldsymbol{Y}_i^*$ can be written

$$\boldsymbol{Y}_i^* = \text{Vec}\left((\boldsymbol{\mathcal{U}}_i^*)^\top\right),$$

where

$$\boldsymbol{\mathcal{U}}_i^* = \left[\boldsymbol{Y}_{i,1}^* \; \boldsymbol{Y}_{i,2}^* \cdots \boldsymbol{Y}_{i,p_s}^*\right]^\top$$

for $\tilde{n} \times 1$ vectors $\boldsymbol{Y}_{i,j}^*$, $j = 1, \ldots, p_s$.

Using (8.4.4), the $\tilde{n}$ columns of the matrix

$$\boldsymbol{M}_{u,n}^{*,i} = \boldsymbol{K}^s \boldsymbol{\mathcal{U}}_i^*$$

are samples from the calibrated emulator on the standardized scale, corresponding to the input settings $\widetilde{\boldsymbol{x}}_1^p, \widetilde{\boldsymbol{x}}_2^p, \ldots, \widetilde{\boldsymbol{x}}_{\tilde{n}}^p$. In the functional case, these samples are transformed back to the original scale using (8.4.1) and (8.4.2) as follows:

$$\boldsymbol{M}_u^{*,i} = \varsigma^s \boldsymbol{M}_{u,n}^{*,i} + \overline{\boldsymbol{y}}^s \boldsymbol{1}_{\tilde{n}}^\top . \tag{8.5.69}$$

This transformation is modified slightly in the multivariate case by using (8.4.3) in place of (8.4.2) producing

$$\boldsymbol{M}_u^{*,i} = \left(\boldsymbol{C}_d^s\right)^{1/2} \boldsymbol{M}_{u,n}^{*,i} + \overline{\boldsymbol{y}}^s \boldsymbol{1}_{\tilde{n}}^\top .$$

Upon completion of the sampling process, N_{mcmc} samples from the calibrated emulator will have been generated for each of the $\tilde{n}$ input settings.

Example 1.7 *(Continued)*. Consider Case 3 emulation. Figure 8.26 shows the centered data for $n = 1$ experiment, superimposed on the $m = 128$ centered simulation runs, and $N_{mcmc} = 1000$ realizations of the centered, calibrated emulator. Using (8.5.69) and noting that $\tilde{n} = 1$, the i^{th} realization of the centered, calibrated emulator is given by the vector $\boldsymbol{M}_u^{*,i} - \overline{\boldsymbol{y}}^s$.

Although the velocity profile of the calibrated emulator demonstrates considerable similarity to that of the experimental data, it is clear that simulator bias is significant relative to uncertainty in the calibrated emulator over large portions of the time domain. Comparing the velocity profiles of the calibrated emulators plotted in Figs. 8.26 and 8.23, the assumption of no simulator discrepancy results in the calibration of $\boldsymbol{\Theta}$ to more closely match experimental velocities at times corresponding to greatest variability (approximately 1–1.2 µs and after 2 µs) than in Case 2 calibration. This solution is compensated for by sacrificing prediction accuracy in the flat portion of the velocity profile. In general, when substantial differences in calibrated emulator performance emerge from comparing the Cases 2 and 3 analyses, attempts should be made to understand their source by examining the simulator, the experimental data, and all statistical assumptions for potential inadequacies. ♦

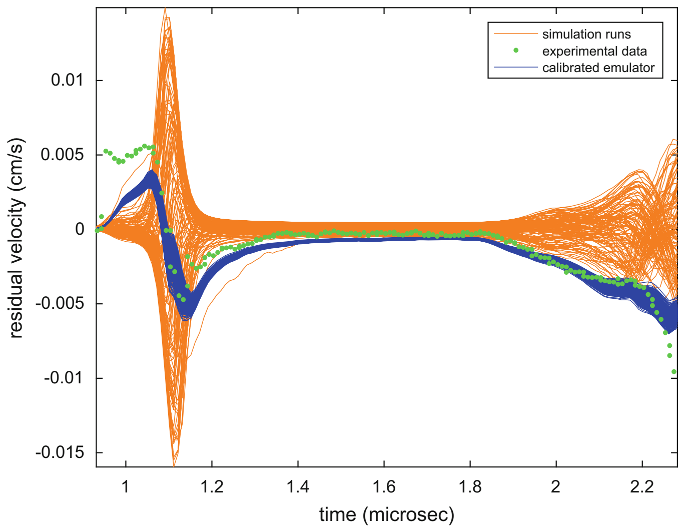


Fig. 8.26 Centered simulation runs (orange lines), centered experimental data (green dots), and centered, calibrated emulator realizations (blue lines)

Predicted values of expected experimental output for the i^{th} sample in this scenario are provided by the calibrated emulator $\boldsymbol{M}_u^{*,i}$. Predicted values of experimental output itself for the i^{th} sample are generated by sampling observation errors according to (8.5.66) and the associated discussion, collecting these errors into the $m_s \times \tilde{n}$ matrix $\mathcal{E}_i^*$, and adding this matrix to the predicted values of expected experimental output,

$$\boldsymbol{Y}_p^{*,i} = \boldsymbol{M}_u^{*,i} + \mathcal{E}_i^* \,.$$

8.6 Chapter Notes

8.6.1 Special Cases of Functional Emulation and Prediction

This subsection treats two common special cases of the KOH model extension to functional outputs presented in Sects. 8.4 and 8.5.

- In some applications, it is not necessary to model simulator basis noise $\boldsymbol{\varepsilon}^s$. This is the case for any setting in which $p_s \le \min\{m_s, m-1\}$ simulator basis vectors are selected and *all* singular values from the SVD of $\boldsymbol{Y}^{s,n}$ in Sect. 8.4.1 starting with the $(p_s + 1)^{st}$ largest take the value 0. In particular, this pertains to multivariate applications where $p_s = m_s$ is chosen. The log prior and log likelihood calculations of Sect. 8.5.1 carry over with the following modifications:

1. Delete the term involving $\lambda_{s,\varepsilon}$ from the log prior distributions (8.5.13), (8.5.14), and (8.5.34).
2. Replace $\boldsymbol{\Sigma}_{W\varepsilon}$ with $\boldsymbol{\Sigma}_W$ from (8.4.24) in every term of the log likelihood functions (8.5.26), (8.5.31), and (8.5.35).

The log posterior distributions (8.5.32) and (8.5.33) that result from applying these modifications are exact when the fixed ridge parameter r_0 from (8.4.37) or (8.4.56) takes the value zero. When $r_0 > 0$, the log likelihood functions (8.5.26) and (8.5.31) must be corrected by additional additive terms involving r_0 to obtain exact log likelihood calculations in this case. However, these corrections are typically assumed to be negligible and ignored. The log posterior distribution (8.5.36) is exact with these modifications applied.

- The mean vectors (8.5.48), (8.5.59), and (8.5.67) and the covariance matrices (8.5.49), (8.5.62), and (8.5.68), of the Cases 1–3 conditional predictive distributions, respectively, are modified when simulator basis noise is assumed negligible by universally replacing $\boldsymbol{\Sigma}_{W\varepsilon}$ with $\boldsymbol{\Sigma}_W$.
- The KOH model for scalar output treated in Sects. 8.1–8.3 can be derived from the KOH model for functional output with some modifications. The model of simulation data sets $m_s = p_s = 1$, and therefore $\boldsymbol{K}^s = 1$ in (8.4.9). The model of experimental data sets $m_{p,i} = 1$ and $p_\delta = 1$, implying $\boldsymbol{K}_i^s = \boldsymbol{D}_i^p = 1$ in (8.4.18) for $i = 1, \ldots, n$. The precision matrix $\lambda_{p,\varepsilon}\boldsymbol{P}_i^p$ of (8.4.12) reduces to a scalar quantity $\lambda_{p,\varepsilon}p_i^p$ for $i = 1, \ldots, n$. The approach of simulator and experimental data reduction via least squares with $r_0 = 0$ reproduces the original data in the case of scalar output. When simulator bias is modeled, the least squares procedure implemented without modification would lead to the $\boldsymbol{F}_{DK}$ matrix defined above (8.4.28) having reduced column rank by construction. This is avoided by setting $\boldsymbol{F}_{DK} = \boldsymbol{I}_n$ and adopting an alternative definition of $\boldsymbol{Z}_\mathrm{d}$ in (8.4.22), $\boldsymbol{Z}_\mathrm{d} = \left(\left(\boldsymbol{V}_p + \boldsymbol{U}_s\right)^\top, \boldsymbol{W}_s^\top\right)^\top$, where $\boldsymbol{V}_p$ and $\boldsymbol{U}_s$ are $n \times 1$ column vectors and $\boldsymbol{W}_s$ is a $m \times 1$ column vector. Its covariance matrix $\boldsymbol{\Sigma}_{Z,\mathrm{d}}$ from (8.4.23) takes the form

$$\boldsymbol{\Sigma}_{Z,\mathrm{d}} = \begin{pmatrix} \boldsymbol{\Sigma}_V + \boldsymbol{\Sigma}_U & \boldsymbol{\Sigma}_{UW} \\ \boldsymbol{\Sigma}_{UW}^\top & \boldsymbol{\Sigma}_W \end{pmatrix} .$$

Summing the discrepancy and calibrated simulator coefficients in the scalar output case implies using $p_\delta + p_s = 1$ instead of 2 when computing quantities in matrix blocks corresponding to the experimental data, resulting in $\boldsymbol{X}_{p,\mathrm{d}} = \boldsymbol{I}_n$ and $\boldsymbol{X}_s = \boldsymbol{I}_m$ in (8.4.26). Using the redefined $\boldsymbol{Z}_\mathrm{d}$ above, the matrix $\boldsymbol{\Sigma}_{\mathrm{d}|W}$ from (8.5.22) becomes

$$\boldsymbol{\Sigma}_{\mathrm{d}|W} = \boldsymbol{\Sigma}_V + \boldsymbol{\Sigma}_{U|W} + \lambda_{p,\varepsilon}^{-1}\left(\boldsymbol{P}^{p,n}\right)^{-1} .$$

The vector $\mathrm{Vec}\left(\left[\widetilde{\boldsymbol{V}}_p\ \widetilde{\boldsymbol{T}}_{\mathrm{d},s}\right]\right)$ in (8.5.26) is replaced by $\boldsymbol{Y}^{p,n} - \boldsymbol{\Sigma}_{UW}\boldsymbol{\Sigma}_{W\varepsilon}^{-1}\boldsymbol{Y}^{s,n,\varepsilon}$. When simulator bias is assumed negligible, the pertinent modifications mentioned above are made and $\rho_\delta = 0$. Reduction to the scalar case carries through without redefining any additional quantities. This is also true when emulation of the simulator is desired based only on simulation output. Calculation of the log posterior distributions (8.5.32), (8.5.33), and (8.5.36) follows by incorporating the relevant adjustments described above. Simulator basis noise is extraneous in the scalar output case and is defined to be identically zero. Therefore, the adjustments described in the first item are also applied when computing these log posterior distributions. Because the least squares procedure reproduces the original data for scalar output, no corrections to the prior distribution of observational error precision are required when computing the log posterior distributions for Case 2 or 3: $a_{\mathrm{d},p,\varepsilon} = a_{\mathrm{nod},p,\varepsilon} = a_{p,\varepsilon}$ and $b_{\mathrm{d},p,\varepsilon} = b_{\mathrm{nod},p,\varepsilon} = b_{p,\varepsilon}$.

- Prediction in the scalar output case as presented in Sect. 8.3 follows from the functional case by first applying the pertinent modifications of the previous item. In Case 2, changes to $\boldsymbol{\Sigma}_{Z^*Z}$ from (8.5.54) and $\boldsymbol{\Sigma}_{Z^*VU|W}$ from (8.5.57) are also required,

$$\boldsymbol{\Sigma}_{Z^*Z} = \begin{bmatrix} \boldsymbol{\Sigma}_{V^*V} & \mathbf{0}_{\tilde{n},m} \\ \boldsymbol{\Sigma}_{U^*U} & \boldsymbol{\Sigma}_{U^*W} \end{bmatrix} \quad \text{and} \quad \boldsymbol{\Sigma}_{Z^*VU|W} = \begin{bmatrix} \boldsymbol{\Sigma}_{V^*V} \\ \boldsymbol{\Sigma}_{U^*U|W} \end{bmatrix}.$$

Cases 1 and 3 carry through without additional modification. The adjustments of the second item must also be made in each of the three cases as simulator basis noise is eliminated for scalar output.

8.6.2 Some Other Perspectives on Emulation and Calibration

This subsection highlights several recent approaches to emulation and calibration of simulators.

Conti and O'Hagan (2010) introduced a Bayesian emulator of functional simulator output possessing a comparable computational burden to that required by the emulators of scalar simulator output developed in Chap. 4. Such efficiency is achieved by assuming the m_s-variate GP model of functional simulator output $\boldsymbol{Y}^s(\boldsymbol{x}^s, \boldsymbol{t})$ has stationary, *separable* covariance structure,

$$Cov\left[\boldsymbol{Y}^s(\boldsymbol{x}_1^s, \boldsymbol{t}_1), \boldsymbol{Y}^s(\boldsymbol{x}_2^s, \boldsymbol{t}_2)\right] = R_s\left((\boldsymbol{x}_1^s, \boldsymbol{t}_1), (\boldsymbol{x}_2^s, \boldsymbol{t}_2)\right)\boldsymbol{\Sigma},$$

for $R_s(\cdot,\cdot)$ a correlation function such as that specified by (8.2.1) and $\boldsymbol{\Sigma}$ a general symmetric, positive definite matrix. As a specific example, the simulator bias model (8.4.16) is separable (with $\boldsymbol{\Sigma} = \boldsymbol{D}^p\,(\boldsymbol{D}^p)^\top$) when a single discrepancy basis group ($F = 1$) is specified. The separability assumption is restrictive, implying that the correlation structure with respect to inputs $(\boldsymbol{x}^s, \boldsymbol{t})$ *separates* from the correlation between each scalar output element of the functional response. This *may* be appropriate for simulators such as the flyer plate code that produce outputs over a space–time index variable mesh, if the dominant physics of the simulator as expressed through input variations has a consistent effect on the output across the entire mesh. Separability should be viewed with caution in multivariate applications where the individual outputs represent different physical quantities.

Overstall and Woods (2016) developed a general class of multivariate emulators that include the Conti and O'Hagan (2010) emulator and the *lightweight* (linear regression-based) emulator of Rougier (2007) as special cases. These emulators are constructed to have separable covariance structure. Recognizing the potential limitations of this restriction, extensions of the Bastos and O'Hagan (2009) diagnostics to the multivariate output setting were proposed for assessment of emulator quality. Furthermore, Bayesian model comparison is used to facilitate selection of an appropriate linear regression mean function. Unordered categorical inputs are accommodated in the GP emulators arising from this framework by assuming the power exponential correlation function (2.2.11) for continuous inputs and multiplying it by a function that is exchangeable in the levels of the categorical inputs. Specifically, the portion of the correlation function contributed by the $d_1 \times 1$ vector of categorical inputs $\boldsymbol{x}^q$ is

$$R_q(\boldsymbol{x}_1^q, \boldsymbol{x}_2^q \,|\, \boldsymbol{\xi}_q) = \exp\left\{ -\sum_{i=1}^{d_1} \xi_{q,i}\, I\{x_{1,i}^q \neq x_{2,i}^q\} \right\},$$

where $I\{\cdot\}$ denotes the indicator function and $\xi_{q,i} \geq 0$ for $i = 1, \ldots, d_1$. This approach or any of the alternatives discussed in Sect. 2.4 can also be invoked to incorporate categorical inputs into the emulation and calibration methodology of Sects. 8.4 and 8.5.

Fricker et al. (2013) discuss emulators of multivariate simulator output based on more flexible nonseparable covariance structures. They recommend two basic approaches to constructing nonseparable covariance functions. The first approach is to convolve smoothing kernels with Gaussian white noise processes (see Sect. 2.5.3); the second approach is to use variations of the nonseparable linear model of coregionalization (NLMC; see Sect. 2.5.2). For example, models (8.4.5) for the simulator and (8.4.16) for the simulator bias are versions of the NLMC approach where the coefficient processes are not restricted to have unit variance; the latter allows additional flexibility for the between-output covariance matrices.

Paulo et al. (2012) took a NLMC approach to Bayesian calibration of computationally expensive simulators that have considerable similarities to the methodology of this chapter when applied to multivariate output (i.e., $p_s = m_s$ and $\boldsymbol{D}^p = \boldsymbol{I}_{m_s}$). Notable differences include taking $\boldsymbol{K}^s = \boldsymbol{U}^s$ in place of (8.4.4), estimating a constant mean for each output via maximum likelihood (ML), replacing the parameters of $\boldsymbol{\Omega}^s$ from (8.4.10)—excluding the nugget effect precisions—by their ML estimates, and adopting a general symmetric, positive definite matrix for the covariance of the experimental data in place of the more restrictive form (8.4.12) under the presumption that sufficient replicate data is available to infer this matrix. This calibration methodology restricts the collection of experimental data to a single setting of the control variables $\boldsymbol{x}^p$ (as was the case with the flyer plate experiment).

The framework of Paulo et al. (2012) allows for multiple sources of experimental data and simulation models to inform on a common set of calibration parameters, as does the methodology of this chapter. Specifically, suppose data $\boldsymbol{Y} = \{\boldsymbol{Y}_1, \boldsymbol{Y}_2, \ldots, \boldsymbol{Y}_L\}$ is available from L sources, where $\boldsymbol{Y}_i$ contains physical observations *and* simulation output corresponding to the i^{th} source. Denote the set of calibration parameters appearing across all sources by $\boldsymbol{\Theta}$, and let $\boldsymbol{\Theta}_i$ denote the subset of $\boldsymbol{\Theta}$ relevant to simulating the i^{th} experiment. Let $\zeta = \{\zeta_1, \zeta_2, \ldots, \zeta_L\}$ collect nuisance parameters (quantities that appear in the likelihood function but not in the simulator, e.g., covariance parameters) across all sources. Assuming conditional independence of data across sources given $\boldsymbol{\Theta}$, the posterior distribution of $(\boldsymbol{\Theta}, \zeta)$ is given by

$$\pi(\boldsymbol{\theta}, \zeta \,|\, \boldsymbol{Y}) \propto \pi(\boldsymbol{\theta}) \prod_{i=1}^{L} L_i(\boldsymbol{\theta}_i, \zeta_i \,|\, \boldsymbol{Y}_i) \prod_{i=1}^{L} p_i(\zeta_i)$$

assuming prior independence of $\boldsymbol{\Theta}$ and ζ, and the components of ζ are mutually independent. Here $L_i(\boldsymbol{\theta}_i, \zeta_i \,|\, \boldsymbol{Y}_i)$ and $p_i(\zeta_i)$ refer to the likelihood function of all parameters and the prior distribution of the nuisance parameters for the i^{th} source, respectively, and $\pi(\boldsymbol{\theta})$ denotes the prior distribution of the calibration parameters. For any given source, the log likelihood function and log prior distribution are provided in Sect. 8.5, noting that only a single log prior distribution of the calibration parameters is used in place of the source-specific log prior distributions of these parameters.

Storlie et al. (2015) introduced the Bayesian smoothing spline (BSS) analysis of variance (ANOVA) methodology (BSS-ANOVA) for emulation and calibration of multivariate output from computationally expensive simulators. The basis of this approach is the Sobol´ decomposition (7.4.9) of each simulator response function. This decomposition is truncated for tractability, and each component remaining is modeled as a mean-zero GP. For continuous inputs, the *BSS-ANOVA covariance function* is adopted for each main effect component of the decomposition. Covariance functions for two-factor and higher-order interactions are obtained as products of the BSS-ANOVA covariance functions. Unordered categorical inputs are handled in a similar fashion using a covariance function that enforces the ANOVA sum-to-zero constraint. This construction results in each of the component processes satisfying the constraints (7.4.13) and (7.4.14).

A BSS-ANOVA surrogate for each simulator response function is constructed by invoking the Karhunen–Loéve theorem to represent each component process as a truncated sum of eigenfunctions weighted by coefficients having independent and identically distributed mean-zero normal distributions. For each eigenfunction of each component process, a general, positive definite covariance matrix is assumed for the vector of corresponding coefficients collected across outputs. This results in a multivariate emulator of the simulator having nonseparable covariance structure. For calibration, a similar multivariate BSS-ANOVA emulator is adopted for the simulator bias functions of the KOH model, and MCMC is used to sample from the joint posterior disribution of the calibration parameters $\boldsymbol{\Theta}$ and other unknowns. BSS-ANOVA scales linearly in the number of simulation runs, allowing it to be used in applications characterized by a large amount of simulation data that would present difficulties for the approach of Sect. 8.5.

Pratola and Hidgon (2016) extended the Bayesian Additive Regression Tree (BART) nonparametric function estimation methodology to enable simulator calibration in the possible presence of systematic bias. BART, equipped with a modified error model, is fit to all the physical observations and simulator output simultaneously, with an addition to the set of input variables (index, control, and calibration) to include an indicator variable of whether a record is a physical observation or simulator output. A split on this indicator will only occur in one or more trees if the simulator exhibits bias relative to the physical observations. This formulation allows discrepancies to be modeled *locally*. That is, this modeling approach can detect if model bias only pertains in a localized subset of the input variable domain, and then provide inference as to its size. This approach also allows for straightforward incorporation of categorical inputs, and does not require the initial specification of a dimension-reducing basis representation such as that utilized in Sect. 8.4, as the basis representation is automatically learned as part of the BART methodology.

Francom et al. (2018) developed methodology for emulating functional simulator output that also achieves significant computational savings relative to the approach of Sect. 8.5 for large data sets of simulation output (e.g., $\geq 10^3$ simulator runs). A statistical model similar to (8.4.8) is adopted that has a modified treatment of simulator basis noise. Essentially, each basis coefficient $W_i^s(\cdot)$ is modeled using adaptive splines that are fit to training data $\widetilde{\boldsymbol{W}}_s^i$ from (8.4.46), where the superscript i indicates the i^{th} column of $\widetilde{\boldsymbol{W}}_s$. Specifically, each basis coefficient is modeled using BMARS (Bayesian Multivariate Adaptive Regression Splines) extended to allow for unordered categorical inputs. Predictions of simulation output with uncertainty are obtained by combining realizations of the basis coefficients applied to their respective eigenfunctions with realizations of the simulator basis noise. Calibration is conducted by first fitting a BMARS model to an estimate of simulator bias calculated by subtracting simulation output generated at the prior mean setting of $\boldsymbol{\Theta}$ from the physical observations and then using MCMC assuming fixed functional forms for the emulator and discrepancy from their respective BMARS fits to sample from the posterior of $\boldsymbol{\Theta}$, observational error variance, and a scale adjustment applied to the fixed discrepancy function.

Bliznyuk et al. (2008) approached Bayesian calibration of computationally expensive simulators from a different perspective than that taken in this chapter. They developed a surrogate model for the log posterior distribution of calibration parameters $\boldsymbol{\Theta}$ and nuisance parameters $\boldsymbol{\zeta}$ itself, *rather than* approximating this distribution through an emulator model of the scalar, multivariate, or functional simulator output. Their algorithm first estimates the posterior mode of $(\boldsymbol{\Theta}, \boldsymbol{\zeta})$ using a derivative-free optimization algorithm. Next the log posterior distribution is estimated by a fitted quadratic surface in the neighborhood of the posterior mode. This surface is used to identify an approximate highest posterior density (HPD) region over which radial basis functions (RBF) are employed to more accurately approximate the log posterior surface. Quantities of interest for prediction are also generated at each training run of the simulator conducted during this process. Finally, MCMC is carried out on the RBF surrogate for the log posterior distribution restricted to the approximate HPD region. Prediction with uncertainty is carried out by propagating MCMC samples of $(\boldsymbol{\Theta}, \boldsymbol{\zeta})$ through a surrogate for the prediction quantities of interest developed from the aforementioned training runs.

8.6.3 Software for Calibration and Validation

This subsection provides links to several software packages that can be used for emulation and calibration of simulators.

- GPMSA (Gaussian Process Models for Simulation Analysis) fits GP-based surrogate models for emulation, sensitivity analysis, and calibration of simulators that produce scalar, multivariate, or functional output. The approach to statistical inference described in Sects. 8.4 and 8.5 is implemented in GPMSA, which was used to analyze the simulation and observational outputs from the flyer plate experiment. GPMSA can be obtained from http://go.osu.edu/GPMSA.
- Dakota is a multilevel parallel object-oriented software package for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis. Dakota supports emulation, sensitivity analysis, and calibration of simulators via several surrogate-based approaches. Dakota also maintains multiple frameworks that facilitate the direct probing of faster simulators within sensitivity analysis and calibration analyses. Dakota can be obtained from Sandia National Laboratories at http://dakota.sandia.gov.
- QUESO (Quantification of Uncertainty for Estimation, Simulation, and Optimization) is a C++ library containing a collection of algorithms and other functionalities aimed at the solution of statistical inverse problems, the solution of statistical forward problems, the validation of a model, and for the prediction of quantities of interest from such a model along with the quantification of their uncertainties. QUESO can be obtained from the University of Texas at Austin *Center for Predictive Engineering and Computational Science* at http://pecos.ices.utexas.edu/software.
- BACCO (Bayesian Analysis of Computer Code Output) is an R package implementation of the KOH model.
- SAVE (Simulator Analysis and Validation Engine) is an R package implementing Bayesian statistical methodology for the analysis of complex computer models. It allows for the emulation, calibration, and validation of computer models following the approach of Bayarri et al. (2007).
- BSS-ANOVA-UQ (Bayesian Smoothing Spline Analysis of Variance) is a file of R functions that implement the KOH model using BSS-ANOVA emulators of multivariate simulation output and simulator bias functions, following the approach of Storlie et al. (2015). BSS-ANOVA-UQ can be obtained from https://drive.google.com/file/d/0B-yrHHWZVJPSTFRrc2ZKeVpZblE/edit.
- calibart is a C/C++ implementation of the BART calibration model and R wrapper functions for use within an R session, following the approach of Pratola and Hidgon (2016). Software available for download at http://www.matthewpratola.com/wp-content/uploads/2015/04/calibart.zip

- BASS (Bayesian Adaptive Spline Surface) is an R package for fitting Bayesian adaptive spline surfaces and performing global sensitivity analyses of these models following the approach of Francom et al. (2018).
- SmartUQ is a commercial package that performs calibration. See https://www.smartuq.com to find out more about the features of SmartUQ.

Appendix A
List of Notation

A.1 Abbreviations

ARD	–	Average reciprocal distance (design) (Sect. 5.4)
BLUP	–	Predictor having minimum mean squared prediction error in the class of predictors that are linear and unbiased (with respect to some family of distributions) (Sect. 3.1)
ERMSPE	–	Empirical root mean squared prediction error (Sect. 3.4)
FE	–	Finite element
GP	–	Gaussian process; Gaussian random function (Sect. 2.2)
i.i.d.	–	Independent and identically distributed
IMSPE	–	Integrated mean squared prediction error (Sect. 6.2)
JE	–	Joint effect (plot or SI)
LHD	–	Latin hypercube design (Sect. 5.2)
LUP	–	Linear unbiased predictor (Sect. 3.2)
MCMC	–	Markov chain Monte Carlo
ME	–	Main effect (plot or SI)
MLE	–	Maximum likelihood estimator
MMSPE	–	Maximum mean squared prediction error (Sect. 6.2)
MmLHD	–	Maximin Latin hypercube design (Sect. 5.4)
mARD	–	Minimum ARD design (Sect. 5.4)
MSPE	–	Mean squared prediction error (Sect. 3.2)
REML	–	Restricted (or residual) maximum likelihood (estimator) (Sect. 3.3)
RMSPE	–	Root mean squared prediction error
SI	–	Sensitivity index
TE	–	Total effect (plot or SI)
XV	–	Cross-validation (Sect. 3.3)

A.2 Symbols

$\mathbf{0}_{r,c}$	–	$r \times c$ matrix of zeros
$\mathbf{1}_n$	–	$n \times 1$ (column) vector of ones
$\setminus$	–	Set difference
$(a)^+$	–	$\max\{a, 0\}$ for $a \in \mathbb{R}$
$(a)^-$	–	$\min\{a, 0\}$ for $a \in \mathbb{R}$
$\lceil a \rceil$	–	Smallest integer greater than or equal to a for $a \in \mathbb{R}$
$\ell n(a)$	–	Natural logarithm of a, $a > 0$
$\lvert a \rvert$	–	Absolute value of a for $a \in \mathbb{R}$
$Be(\alpha,\beta)$	–	The beta distribution with probability density (B.3.1) (Appendix B.3)
$\binom{n}{j}$	–	$n!/(j!(n-j)!)$, for integer j with $0 \le j \le n$ is the number of subsets of size j that can be drawn from n distinct objects
$Cov[Y(\boldsymbol{x}_1), Y(\boldsymbol{x}_2)]$	–	Process model covariance of $Y(\cdot)$
$D_\infty(\mathcal{D})$	–	Star discrepancy (from the uniform distribution) given by (5.6.3)
$diag(\boldsymbol{a})$	–	Diagonal matrix with elements $a_1, \ldots, a_m$, where $\boldsymbol{a} = (a_1, \ldots, a_m)$
$\det(\boldsymbol{W})$	–	Determinant of the square matrix $\boldsymbol{W}$
$\xi^\alpha(\boldsymbol{x}_c)$	–	Upper α quantile of the distribution of $y^s(\boldsymbol{x}_c, \boldsymbol{X}_e)$ induced by random environmental variables $\boldsymbol{X}_e$ for fixed control variable $\boldsymbol{x}_c$ (Sect. 1.3.2)
$e(\boldsymbol{x}_i)$	–	The i^{th} standardized residual based on data $y(\boldsymbol{x}_1), \ldots, y(\boldsymbol{x}_n)$ (so that $e(\boldsymbol{x}_1), \ldots, e(\boldsymbol{x}_n)$ have sample mean zero and unit sample variance)
$\boldsymbol{e}_i$	–	The i^{th} unit column vector $(0, \ldots, 0, 1, 0, \ldots, 0)^\top$ where 1 is in the i^{th} position
$E[\cdot]$	–	Expectation with respect to the process model under consideration
$f_j(\cdot)$	–	The j^{th} regression function in the mean of a stochastic process model for $Y(\cdot)$
$\Gamma(\alpha,\beta)$	–	The gamma distribution with probability density (B.2.1) which has mean α/β and variance α/β^2 (Appendix B.2)
$GP(\boldsymbol{\mu}, \lambda_z, \cdot)$	–	Gaussian process with mean $\boldsymbol{\mu}$, precision λ_z, and either a specified correlation function $R(\cdot)$ or the parameters of a specified parametric correlation family
$\boldsymbol{I}_n$	–	$n \times n$ identity matrix
$I\{E\}$	–	Indicator function of the event E which is defined to be 1 or 0 as E is true or false
$\boldsymbol{J}_n$	–	$n \times n$ matrix of ones, i.e., $\boldsymbol{J}_n \equiv \mathbf{1}_n \mathbf{1}_n^\top$
$-k$	–	The set $\{1, \ldots, d\} \setminus k$ when d is a given positive integer and $k \in \{1, \ldots, d\}$

$\boldsymbol{\kappa}$	–	Vector of all unknown parameters in a correlation function
m	–	Number of outputs in a multiple response application (Sect. 1.3.3)
$\mu(\boldsymbol{x}_c)$	–	Mean of $y^s(\boldsymbol{x}_c, \boldsymbol{X}_e)$ induced by random environmental variables $\boldsymbol{X}_e$ for fixed control variable $\boldsymbol{x}_c$ (Sect. 1.3.2)
n_e	–	Number of inputs at which predictions are to be made
n_p	–	Number of training runs available in a physical experiment
n_s	–	Number of training runs available from a simulator experiment
$N(\mu, \sigma^2)$	–	The univariate normal distribution with mean μ and variance σ^2
$N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	–	The p-dimensional multivariate normal distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ (Appendix B.1)
$\boldsymbol{A} \otimes \boldsymbol{B}$	–	Kronecker product of $\boldsymbol{A}$ and $\boldsymbol{B}$
$-Q$	–	Given a positive integer d, the set difference $\{1, \ldots, d\} \setminus Q$ when $Q \subset \{1, \ldots, d\}$
ρ_j	–	Correlation parameter in a Gaussian correlation model
$R(\cdot)$	–	Correlation function (Sect. 2.2.2)
$\boldsymbol{R}$	–	$n \times n$ matrix of pairwise correlations of $\boldsymbol{Y}^n$ elements (Sect. 3.2)
$\mathbb{R}$	–	Set of real numbers
$\mathrm{rank}(\boldsymbol{W})$	–	Rank of the $r \times c$ matrix $\boldsymbol{W}$
$s(\boldsymbol{x}^{te})$	–	Root MSPE of $\widehat{y}(\boldsymbol{x}^{te})$, the square root of $E\left[(Y(\boldsymbol{x}^{te}) - \widehat{y}(\boldsymbol{x}^{te}))^2\right]$
$\bullet^\top$	–	Transpose of a vector or matrix
$\mathrm{tr}(\boldsymbol{W})$	–	Trace of the square matrix $\boldsymbol{W}$
$\boldsymbol{\theta}$	–	True values of the (unknown) model inputs to a simulator (Sect. 8.2)
$\boldsymbol{\Theta}$	–	Random variable describing the prior uncertainty for $\boldsymbol{\theta}$ (Sect. 8.2)
T_ν	–	The central univariate t distribution with ν degrees of freedom
t_ν^α	–	Upper α quantile of the central univariate t distribution with ν degrees of freedom, i.e., $P\{W \geq t_\nu^\alpha\} = \alpha$ where $W \sim T_\nu$
$T_p(\nu, \boldsymbol{\mu}, \boldsymbol{\Sigma})$	–	The p-dimensional non-central multivariate student t distribution with ν degrees of freedom, location vector $\boldsymbol{\mu} \in \mathbb{R}^p$, and positive definite scale matrix $\boldsymbol{\Sigma}$ (Appendix B.4)
$T_1(\nu, \mu, \sigma)$	–	The non-central univariate student t distribution with ν degrees of freedom, location $\mu \in \mathbb{R}$, and scale parameter $\sigma > 0$, a special case of the $T_p(\nu, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ distribution (Appendix B.4)
$U(a, b)$	–	The uniform distribution over the interval (a, b)
$\mathrm{Vec}(\boldsymbol{A})$	–	The $rc \times 1$ column vector obtained from the $r \times c$ matrix $\boldsymbol{A}$ having columns $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_c$ by stacking $\boldsymbol{a}_i$ over $\boldsymbol{a}_{i+1}$ for $i = 1, \ldots, c-1$

$\boldsymbol{\vartheta}$	–	Vector of all unknown parameters in a GP model (including all unknown mean, variance/precision, and correlation parameters)
$\boldsymbol{x}_c$	–	Vector of control (engineering) variables (Sect. 1.3.1)
$\boldsymbol{x}_e$	–	Vector of environmental (noise) variables (Sect. 1.3.1)
$\boldsymbol{x}_m$	–	Vector of model inputs to a computer simulator (Sect. 1.3.1)
$\boldsymbol{x}$	–	Vector of *all* input variables used by a given computer simulator including whatever $\boldsymbol{x}_c$, $\boldsymbol{x}_e$, and $\boldsymbol{x}_m$ are required
$\mathcal{X}$	–	Sample space for the vector of all input variables $\boldsymbol{x}$ (Sect. 1.3)
$\boldsymbol{\xi}$	–	Vector of *rate* (or more descriptively, "roughness") parameters in the Gaussian or other correlation function
$\boldsymbol{x}^{te}$	–	Vector of inputs at which a prediction, such as $y^s(\boldsymbol{x}^{te})$, is to be made
$\boldsymbol{x}_i^{tr}$	–	i^{th} training data vector of inputs
$y^s(\cdot)$	–	The output from running a computer simulator
$y^p(\cdot)$	–	The output from conducting a physical experiment
$[\boldsymbol{X} \mid \boldsymbol{Y}]$	–	Conditional distribution of $\boldsymbol{X}$ given $\boldsymbol{Y}$
$\lVert\boldsymbol{v}\rVert_p$	–	$\left(\sum_{i=1}^n \mid v_i \mid^p\right)^{1/p}$, the p-norm of the vector $\boldsymbol{v} \in \mathbb{R}^n$; $p = 1$ is sometimes called the Manhattan or taxicab norm, $p = 2$ is the Euclidean norm
z^α	–	Upper α quantile of the standard normal distribution, i.e., $P\{Z \geq z^\alpha\} = \alpha$ where $Z \sim N(0, 1)$

Appendix B
Mathematical Facts

B.1 The Multivariate Normal Distribution

There are several equivalent ways of defining the multivariate normal distribution. Because we mention both degenerate ("singular") and nondegenerate ("nonsingular") multivariate normal distributions, we will define this distribution by the standard device of describing it indirectly as the distribution that arises by forming a certain function, affine combinations, of independent and identically distributed standard normal random variables.

Definition. Suppose $\boldsymbol{Z} = (Z_1, \ldots, Z_r)^\top$ consists of independent and identically distributed $N(0, 1)$ random variables, $\boldsymbol{L}$ is an $m \times r$ real matrix, and $\boldsymbol{\mu}$ is an $m \times 1$ real vector. Then

$$\boldsymbol{W} = (W_1, \ldots, W_m)^\top = \boldsymbol{L}\boldsymbol{Z} + \boldsymbol{\mu}$$

is said to have the *multivariate normal distribution* (associated with $\boldsymbol{\mu}$, $\boldsymbol{L}$).

It is straightforward to compute the mean vector of $(W_1, \ldots, W_m)$ and the matrix of the variances and covariances of the $(W_1, \ldots, W_m)$ in terms of $(\boldsymbol{\mu}, \boldsymbol{L})$ as

$$E[\boldsymbol{W}] = \boldsymbol{\mu} \ \text{ and } \ \mathrm{Cov}[\boldsymbol{W}] = E[(\boldsymbol{W} - \boldsymbol{\mu})(\boldsymbol{W} - \boldsymbol{\mu})^\top] = \boldsymbol{L}\boldsymbol{L}^\top.$$

As examples, suppose Z_1 and Z_2 are independent $N(0, 1)$. First let

$$\boldsymbol{W} = \begin{pmatrix} W_1 \\ W_2 \end{pmatrix} = \boldsymbol{L}\boldsymbol{Z} + \boldsymbol{\mu} = \begin{bmatrix} 3 & 0 \\ 5 & 0 \end{bmatrix} \begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix} + \begin{pmatrix} 2 \\ 0 \end{pmatrix} = \begin{pmatrix} 3Z_1 + 2 \\ 5Z_1 + 0 \end{pmatrix}.$$

By definition, $\boldsymbol{W}$ has a multivariate normal distribution. However W_1 and W_2 are "singular" in the sense that W_1 and W_2 are linearly dependent via $W_2 = 5(W_1 - 2)/3$. Similarly,

$$\begin{pmatrix} W_1 \\ W_2 \end{pmatrix} = \begin{bmatrix} 3 & 2 \\ 0 & 5 \end{bmatrix} \begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix} + \begin{pmatrix} 2 \\ 0 \end{pmatrix}$$

T. J. Santner et al., *The Design and Analysis of Computer Experiments*, Springer Series in Statistics, https://doi.org/10.1007/978-1-4939-8847-1

also has a multivariate normal distribution. However, in this case W_1 and W_2 are "nonsingular" in the sense of being linearly independent.

This example illustrates the fundamental dichotomy in multivariate normal distributions. A multivariate normal distribution is *nonsingular* (nondegenerate) if the rows of $\boldsymbol{L}$ are linearly independent, i.e, $\text{rank}(\boldsymbol{L}) = m$, and it is *singular* (degenerate) if the rows of $\boldsymbol{L}$ are linearly dependent, i.e, $\text{rank}(\boldsymbol{L}) < m$.

Suppose $\boldsymbol{W}$ has the nonsingular multivariate normal distribution defined by $\boldsymbol{\mu} \in \mathbb{R}^m$ and $m \times r$ matrix $\boldsymbol{L}$ having rank m. Let

$$\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^\top$$

denote the covariance matrix of $\boldsymbol{W}$. Notice that $\boldsymbol{\Sigma}$ must be symmetric and positive definite (the latter follows because if $\|\cdot\|_2$ denotes Euclidean norm and $\boldsymbol{z} \neq \boldsymbol{0}$, then $\boldsymbol{z}^\top\boldsymbol{\Sigma}\boldsymbol{z} = \boldsymbol{z}^\top\boldsymbol{L}\boldsymbol{L}^\top\boldsymbol{z} = \|\boldsymbol{L}^\top\boldsymbol{z}\|_2^2 > 0$ because $\text{rank}(\boldsymbol{L}) = m$). In this case it can be shown that $\boldsymbol{W} = (W_1, \ldots, W_m)$ has density

$$f(\boldsymbol{w}) = \frac{1}{(2\pi)^{m/2}(\det(\boldsymbol{\Sigma}))^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{w}-\boldsymbol{\mu})^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{w}-\boldsymbol{\mu})\right\} \qquad \text{(B.1.1)}$$

over $\boldsymbol{w} \in \mathbb{R}^m$. We denote the fact that $\boldsymbol{W}$ has the nonsingular multivariate normal distribution (B.1.1) by $\boldsymbol{W} \sim N_m(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. There are numerous algorithms for computing various quantiles associated with multivariate normal distributions. We note, in particular, Dunnett (1989) who provides a FORTRAN 77 program for computing equicoordinate percentage points of multivariate normal distributions having product correlation structure (see also Odeh et al. (1988)).

Now suppose $\boldsymbol{W}$ has the singular multivariate normal distribution defined by $\boldsymbol{\mu} \in \mathbb{R}^m$ and $m \times r$ matrix $\boldsymbol{L}$ where $\text{rank}(\boldsymbol{L}) = q < m$. Then $m - q$ rows of $\boldsymbol{L}$ can be expressed as linear combinations of the remaining q rows of $\boldsymbol{L}$, and the corresponding $m - q$ components of $\boldsymbol{W} - \boldsymbol{\mu}$ can be expressed as (the same) linear combinations of the remaining q components of $\boldsymbol{W} - \boldsymbol{\mu}$. Thus, in this case, the support of $\boldsymbol{W}$ is on a hyperplane in a lower-dimensional subspace of $\mathbb{R}^m$. Furthermore, the q components of $\boldsymbol{W}$ used to express the remaining variables have a nonsingular multivariate normal distribution with density on $\mathbb{R}^q$. To illustrate the singular case, consider the toy example above. Marginally, both W_1 and W_2 have proper normal distributions with $W_1 = 3Z_1 + 2 \sim N(2, 9)$ and $W_2 = 5Z_1 \sim N(0, 25)$. Here $m = 2 > 1 = q$. Given either W_1 or W_2, the other random variable can be expressed in terms of the first. For example, given W_1, $W_2 = 5(W_1 - 2)/3$ with probability one, or given W_2, $W_1 = 2 + 3W_2/5$ with probability one.

In the text we make use of the following integration formula, which is an application of the fact that (B.1.1) is a density function.

Lemma B.1. For any $n \times 1$ vector $\boldsymbol{v}$ and any $n \times n$ symmetric, positive definite matrix $\boldsymbol{A}$,

$$\int_{\mathbb{R}^n} \exp\left\{-\frac{1}{2}\boldsymbol{w}^\top\boldsymbol{A}^{-1}\boldsymbol{w} + \boldsymbol{v}^\top\boldsymbol{w}\right\} d\boldsymbol{w} = (2\pi)^{n/2}(\det(\boldsymbol{A}))^{1/2} \exp\left\{\frac{1}{2}\boldsymbol{v}^\top\boldsymbol{A}\boldsymbol{v}\right\}. \qquad \text{(B.1.2)}$$

To prove formula (B.1.2), consider the $N_n(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ multivariate normal density with covariance matrix $\boldsymbol{\Sigma} = \boldsymbol{A}$ and mean $\boldsymbol{\mu} = \boldsymbol{\Sigma}\boldsymbol{v}$. Then

$$\begin{aligned}(2\pi)^{n/2}(\det(\boldsymbol{\Sigma}))^{1/2} &= \int_{\mathbb{R}^n} \exp\left\{-\frac{1}{2}(\boldsymbol{w}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{w}-\boldsymbol{\mu})\right\} d\boldsymbol{w} \qquad \text{(B.1.3)}\\ &= \int_{\mathbb{R}^n} \exp\left\{-\frac{1}{2}\boldsymbol{w}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{w} + \boldsymbol{\mu}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{w} - \frac{1}{2}\boldsymbol{\mu}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right\} d\boldsymbol{w}\,.\end{aligned}$$

Substituting for $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}$ and rearranging terms give the result. □

Perhaps more usefully, we can interpret the proof of Lemma B.1 as stating that if $\boldsymbol{W}$ has density $f(\boldsymbol{w})$, for which

$$f(\boldsymbol{w}) \propto \exp\left\{-\frac{1}{2}\boldsymbol{w}^\top \boldsymbol{A}^{-1}\boldsymbol{w} + \boldsymbol{v}^\top \boldsymbol{w}\right\}, \text{ then } \boldsymbol{W} \sim N_n(\boldsymbol{A}\boldsymbol{v}, \boldsymbol{A})\,. \qquad \text{(B.1.4)}$$

We also require the following result concerning the conditional distribution of a set of components of the multivariate normal distribution given the remaining ones.

Lemma B.2 (Conditional Distribution of the Multivariate Normal). Suppose that

$$\begin{pmatrix}\boldsymbol{W}_1\\ \boldsymbol{W}_2\end{pmatrix} \sim N_{m+n}\left(\begin{pmatrix}\boldsymbol{\mu}_1\\ \boldsymbol{\mu}_2\end{pmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{1,1} & \boldsymbol{\Sigma}_{1,2}\\ \boldsymbol{\Sigma}_{2,1} & \boldsymbol{\Sigma}_{2,2}\end{bmatrix}\right)$$

where $\boldsymbol{\mu}_1$ is $m \times 1$, $\boldsymbol{\mu}_2$ is $n \times 1$, $\boldsymbol{\Sigma}_{1,1}$ is $m \times m$, $\boldsymbol{\Sigma}_{1,2} = \boldsymbol{\Sigma}_{2,1}^\top$ is $m \times n$, and $\boldsymbol{\Sigma}_{2,2}$ is $n \times n$. Then the conditional distribution $[\boldsymbol{W}_1 \mid \boldsymbol{W}_2]$ is

$$N_m\left(\boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{1,2}\boldsymbol{\Sigma}_{2,2}^{-1}(\boldsymbol{W}_2 - \boldsymbol{\mu}_2), \boldsymbol{\Sigma}_{1,1} - \boldsymbol{\Sigma}_{1,2}\boldsymbol{\Sigma}_{2,2}^{-1}\boldsymbol{\Sigma}_{2,1}\right).$$

B.2 The Gamma Distribution

The univariate random variable W is said to have the *gamma* distribution with shape parameter $\alpha > 0$ and rate parameter $\beta > 0$ provided it has probability density function

$$f(w) = \frac{\beta^\alpha}{\Gamma(\alpha)} w^{\alpha-1} e^{-\beta w}, \quad w > 0. \qquad \text{(B.2.1)}$$

This distribution is denoted by $W \sim \Gamma(\alpha,\beta)$; the mean and variance of the $\Gamma(\alpha,\beta)$ distribution are α/β and α/β^2, respectively. An important subfamily of the gamma distributions is the *chi-square* set of distributions; the χ-square with ν degrees of freedom, $\nu \in \{1, 2, \ldots\}$, is the $\Gamma(\nu/2, 1/2)$ distribution which is also denoted by χ^2_ν.

B.3 The Beta Distribution

The univariate random variable W is said to have the *beta* distribution with parameters $\alpha > 0$ and $\beta > 0$ provided it has probability density function

$$f(w) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\,\Gamma(\beta)} w^{\alpha-1}\,(1-w)^{\beta-1}, \quad 0 < w < 1. \tag{B.3.1}$$

This distribution is denoted by $W \sim Be(\alpha,\beta)$; the mean and variance of the $Be(\alpha,\beta)$ distribution are $\alpha/(\alpha+\beta)$ and $\alpha\beta/(\alpha+\beta)^2(\alpha+\beta+1)$, respectively. An important special case of the beta distribution is $\alpha = 1 = \beta$ which yields the *uniform* distribution over (0,1).

B.4 The Non-central Student *t* Distribution

This subsection defines the multivariate (and univariate) Student t distribution. Throughout the section assume that $\boldsymbol{\mu} \in \mathbb{R}^m$ and $\boldsymbol{\Sigma}$ is a positive definite matrix.

Definition. For $m \geq 1$, the random vector $\boldsymbol{W} = (W_1, \ldots, W_m)$ with joint probability density

$$f(\boldsymbol{w}) = \frac{\Gamma((\nu+m)/2)}{(\det(\boldsymbol{\Sigma}))^{1/2}(\nu\pi)^{m/2}\Gamma(\nu/2)} \left(1 + \frac{1}{\nu}(\boldsymbol{w}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{w}-\boldsymbol{\mu})\right)^{-(\nu+m)/2} \tag{B.4.1}$$

for $\boldsymbol{w} \in \mathbb{R}^m$ is said to have the *nonsingular m-variate t distribution* with ν degrees of freedom, location parameter $\boldsymbol{\mu}$, and scale matrix $\boldsymbol{\Sigma}$.

We denote the multivariate t distribution (B.4.1) by $\boldsymbol{W} \sim T_m(\nu, \boldsymbol{\mu}, \boldsymbol{\Sigma})$. The $T_m(\nu, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ distribution has mean vector $\boldsymbol{\mu}$ provided $\nu > 1$ and has covariance matrix $\nu\boldsymbol{\Sigma}/(\nu-2)$ provided $\nu > 2$. Notice that the fact that the joint density (B.4.1) cannot be factored into terms each depending on a single w_i shows that $W_1, \ldots, W_m$ are dependent even if $\boldsymbol{\Sigma} = \boldsymbol{I}_m$. Alternatively, this dependence is clear from the following representation of the m-variate t distribution. If $\boldsymbol{X} \sim N_m(\boldsymbol{0}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma}$ is $m \times m$ symmetric and positive definite and $\boldsymbol{X}$ is independent of $V \sim \nu/\chi^2_\nu$, then

$$\boldsymbol{W} = (W_1, \ldots, W_m) \equiv \left(\sqrt{V}X_1, \ldots, \sqrt{V}X_m\right) \sim T_m(\nu, \boldsymbol{0}_m, \boldsymbol{\Sigma}).$$

The multiplication of each X_i by the common factor $\sqrt{V}$ shows the W_i are dependent. Two other stochastically equivalent ways of defining the m-variate t distribution also provide insight to the meaning of this distribution:

- Suppose that $\boldsymbol{W} \sim T_m(\nu, \boldsymbol{0}_m, \boldsymbol{I}_m)$, $\boldsymbol{\Sigma}$ is positive definite with square root $\boldsymbol{\Sigma}^{1/2}$, and $\boldsymbol{\mu}$ is $m \times 1$, then $\boldsymbol{\mu} + \boldsymbol{\Sigma}^{1/2}\boldsymbol{W} \sim T_m(\nu, \boldsymbol{\mu}, \boldsymbol{\Sigma})$.

- Suppose $\boldsymbol{Z} = (Z_1, \ldots, Z_m)^\top$ has independent and identically distributed $N(0,1)$ components, $\boldsymbol{A}$ is $m \times m$ of rank m, $\boldsymbol{Z}$ is independent of $V \sim \nu/\chi^2_\nu$, and $\boldsymbol{\mu}$ is $m \times 1$, then $\boldsymbol{W} = \boldsymbol{\mu} + \sqrt{V}\boldsymbol{A}\boldsymbol{Z} \sim T_m(\nu, \boldsymbol{\mu}, \boldsymbol{A}\boldsymbol{A}^\top)$.

The "usual" univariate t and unit variance multivariate t distribution are the special cases

- $T_1(\nu, 0, 1)$ and
- $T_m(\nu, \boldsymbol{0}_m, \boldsymbol{R})$

of (B.4.1) where $\boldsymbol{R}$ is positive definite with unit diagonal elements (Odeh et al. (1988) and Kotz and Nadarajah (2004)).

B.5 Some Results from Matrix Algebra

The following formula for the inverse of a 2×2 partitioned matrix can be found as a special case of Theorem 8.5.11 of Harville (1997), among many other sources.

Lemma B.3 (Inversion of a Partitioned Matrix—I). Suppose that $\boldsymbol{B}$ is a nonsingular $n \times n$ matrix and

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{D} & \boldsymbol{A}^\top \\ \boldsymbol{A} & \boldsymbol{B} \end{bmatrix}$$

where $\boldsymbol{D}$ is $m \times m$ and $\boldsymbol{A}$ is $n \times m$. Then $\boldsymbol{T}$ is nonsingular if and only if

$$\boldsymbol{Q} = \boldsymbol{D} - \boldsymbol{A}^\top \boldsymbol{B}^{-1} \boldsymbol{A}$$

is nonsingular. In this case, $\boldsymbol{T}^{-1}$ is given by

$$\begin{bmatrix} \boldsymbol{Q}^{-1} & -\boldsymbol{Q}^{-1}\boldsymbol{A}^\top\boldsymbol{B}^{-1} \\ -\boldsymbol{B}^{-1}\boldsymbol{A}\boldsymbol{Q}^{-1} & \boldsymbol{B}^{-1} + \boldsymbol{B}^{-1}\boldsymbol{A}\boldsymbol{Q}^{-1}\boldsymbol{A}^\top\boldsymbol{B}^{-1} \end{bmatrix}. \tag{B.5.1}$$

That (B.5.1) is $\boldsymbol{T}^{-1}$ can easily be verified by multiplication. To verify the "only if" part of the lemma, see Harville (1997), for example.

The case of Lemma B.3 corresponding to $\boldsymbol{D} = \boldsymbol{0}$ occurs frequently. If $\boldsymbol{B}$ is a nonsingular $n \times n$ matrix and $\boldsymbol{A}$ is $n \times m$, then

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{A}^\top \\ \boldsymbol{A} & \boldsymbol{B} \end{bmatrix}$$

has an inverse if and only if

$$\boldsymbol{A}^\top \boldsymbol{B}^{-1} \boldsymbol{A}$$

is nonsingular. In this case, $\boldsymbol{T}^{-1}$ is given by

$$\begin{bmatrix} -\left(\boldsymbol{A}^\top \boldsymbol{B}^{-1}\boldsymbol{A}\right)^{-1} & \left(\boldsymbol{A}^\top \boldsymbol{B}^{-1}\boldsymbol{A}\right)^{-1}\boldsymbol{A}^\top \boldsymbol{B}^{-1} \\ \boldsymbol{B}^{-1}\boldsymbol{A}\left(\boldsymbol{A}^\top \boldsymbol{B}^{-1}\boldsymbol{A}\right)^{-1} & \boldsymbol{B}^{-1} - \boldsymbol{B}^{-1}\boldsymbol{A}\left(\boldsymbol{A}^\top \boldsymbol{B}^{-1}\boldsymbol{A}\right)^{-1}\boldsymbol{A}^\top \boldsymbol{B}^{-1} \end{bmatrix}.$$

Lemma B.4. Suppose that $\boldsymbol{A}$ is a nonsingular $n \times n$ matrix, $\boldsymbol{C}$ is a nonsingular $m \times m$ matrix, and $\boldsymbol{B}$ is an arbitrary $n \times m$ matrix such that $\left(\boldsymbol{B}^\top \boldsymbol{A}^{-1}\boldsymbol{B} + \boldsymbol{C}^{-1}\right)$ is nonsingular. Then $(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{C}\boldsymbol{B}^\top)$ is a nonsingular $n \times n$ matrix with inverse given by

$$\left(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{C}\boldsymbol{B}^\top\right)^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{B}\left(\boldsymbol{B}^\top \boldsymbol{A}^{-1}\boldsymbol{B} + \boldsymbol{C}^{-1}\right)^{-1}\boldsymbol{B}^\top \boldsymbol{A}^{-1}.$$

Proof: Multiply the right-hand expression by $(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{C}\boldsymbol{B}^\top)$ and verify that it is the identity. □

Two important special cases of Lemma B.4 (corresponding to $\boldsymbol{B} = \boldsymbol{I}_n$) are

$$\begin{aligned}(\boldsymbol{A} + \boldsymbol{C})^{-1} &= \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\left(\boldsymbol{A}^{-1} + \boldsymbol{C}^{-1}\right)^{-1}\boldsymbol{A}^{-1} \\ (\boldsymbol{A} + \boldsymbol{C})^{-1} &= \boldsymbol{C}^{-1} - \boldsymbol{C}^{-1}\left(\boldsymbol{A}^{-1} + \boldsymbol{C}^{-1}\right)^{-1}\boldsymbol{C}^{-1}. \end{aligned} \quad \text{(B.5.2)}$$

Lemma B.5. Suppose that $a \neq 0$ and $b \neq -a/n$, then

- $(a\boldsymbol{I}_n + b\boldsymbol{J}_n)^{-1} = \left(\frac{1}{a}\boldsymbol{I}_n - \frac{b}{a(a+nb)}\boldsymbol{J}_n\right)$ and
- $\det(a\boldsymbol{I}_n + b\boldsymbol{J}_n) = a^{n-1}(a + nb)$

where $\boldsymbol{I}_n$ is the $n \times n$ identity matrix and $\boldsymbol{J}_n$ is the $n \times n$ matrix of ones.

The final concepts discussed in this appendix are those of the *Vec* operator and *vec-permutation* matrices (Sects. 16.2 and 16.3 of Harville (1997) give an in-depth discussion of these notions).

Given an $r \times c$ matrix $\boldsymbol{A}$ with $(r \times 1)$ columns $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_c$, Vec($\boldsymbol{A}$) is the $rc \times 1$ vector obtained by stacking the columns of $\boldsymbol{A}$ in the order $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_c$, i.e.,

$$\text{Vec}(\boldsymbol{A}) = \begin{pmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \vdots \\ \boldsymbol{a}_c \end{pmatrix}.$$

Vec-permutation matrices are used to streamline the derivation of the formulas for the joint statistical model in Sect. 8.4. Given integers $r \geq 1$ and $c \geq 1$, the vec-permutation matrix $\boldsymbol{Q}_{r,c}$ is the $rc \times rc$ permutation matrix that satisfies

$$\text{Vec}(\boldsymbol{A}^\top) = \boldsymbol{Q}_{r,c}\text{Vec}(\boldsymbol{A}) \quad \text{(B.5.3)}$$

for any $r \times c$ matrix $\boldsymbol{A}$. In words, $\boldsymbol{Q}_{r,c}$ permutes the elements of $\boldsymbol{A}$ which are listed in column order in the $rc \times 1$ vector Vec($\boldsymbol{A}$) to be those of $\boldsymbol{A}$ listed in row order, which are the entries of the $rc \times 1$ vector Vec($\boldsymbol{A}^\top$).

It can be shown that $\boldsymbol{Q}_{r,c}$ is the permutation matrix that has $[(i-1)c+j]^{th}$ row equal to the $[(j-1)r+i]^{th}$ row of the $rc \times rc$ identity matrix $\boldsymbol{I}_{rc}$ for $i = 1, \ldots, r$ and $j = 1, \ldots, c$ (Harville (1997)). For example, it is straightforward to calculate that the 6×6 vec-permutation matrix

$$\boldsymbol{Q}_{2,3} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

satisfies (B.5.3) for *any* 2×3 matrix

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}.$$

Three properties of the Vec operator and vec-permutation matrices used in this text are stated next.

1. For any $r_1 \times r_2$ matrix $\boldsymbol{A}$,

$$\text{Vec}(\boldsymbol{A}) = \boldsymbol{Q}_{r_2,r_1}\text{Vec}(\boldsymbol{A}^\top) \tag{B.5.4}$$

for vec-permutation matrix $\boldsymbol{Q}_{r_2,r_1}$.

2. For any $r_1 \times r_2$ matrix $\boldsymbol{A}$, $r_2 \times r_3$ matrix $\boldsymbol{B}$, and $r_3 \times r_4$ matrix $\boldsymbol{C}$,

$$\text{Vec}(\boldsymbol{ABC}) = (\boldsymbol{C}^\top \otimes \boldsymbol{A})\text{Vec}(\boldsymbol{B}). \tag{B.5.5}$$

3. For any $r_1 \times r_2$ matrix $\boldsymbol{A}$ and $r_3 \times r_4$ matrix $\boldsymbol{B}$,

$$(\boldsymbol{A} \otimes \boldsymbol{B})\boldsymbol{Q}_{r_2,r_4} = \boldsymbol{Q}_{r_1,r_3}(\boldsymbol{B} \otimes \boldsymbol{A}) \tag{B.5.6}$$

for vec-permutation matrices $\boldsymbol{Q}_{r_1,r_3}$ and $\boldsymbol{Q}_{r_2,r_4}$.

Appendix C
An Overview of Selected Optimization Algorithms

Optimization problems are required to implement many of the methodologies described in this book. For example, EBLUP prediction assumes that unknown correlation function parameters be estimated, typically by maximizing a likelihood or REML likelihood. Maximizing an expected improvement must be carried out to implement many of the adaptive designs described in Chap. 6. Similarly, designs constructed to optimize the criteria described in Chaps. 5 and 6 require maximization or minimization.

These problems are often difficult because of the nonlinear character of the function to be optimized and because the function can have multiple local modes, and depending on the parameterization, the function's optima are only realized asymptotically.

Because a maximization problem can be turned into a minimization problem by multiplying the objective by minus one, this section considers only algorithms for *minimizing* a given

$$f(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}.$$

The algorithms are often classified as seeking a "global" optimum if $\mathcal{X}$ is a Euclidean space or a "constrained" optimum if $\mathcal{X}$ is a bounded subset of a Euclidean space. Because most minimization algorithms are only guaranteed to find local minima, a common strategy is to use the results of an algorithm that permits random searches of the input domain $\mathcal{X}$, to find many approximate local minima which are used as starting points for runs of a Newton algorithm that can rapidly and accurately find the local minimizer "close" to each given starting value; for example, Butler et al. (2014) use the `DIRECT` algorithm to identify starting points for a gradient search of maximum likelihood estimates of correlation parameters. The summary below focuses on describing the update steps of the algorithms.

C.1 Newton/Quasi-Newton Algorithms

The Newton–Raphson (NR) algorithm is used to find zeros of a given function, say $g(\boldsymbol{x})$. To minimize $f(\boldsymbol{x})$, the NR algorithm is applied to find $\boldsymbol{x}^\star$ satisfying the first-order stationarity condition $f'(\boldsymbol{x}^\star) = 0$.

To illustrate this zero-finding algorithm, suppose that $g(\cdot)$ is a function of a single variable x and $g(x^\star) = 0$. The update step of the algorithm is based on the first-order Taylor approximation

$$g(x) \approx g(x^\star) + g'(x^\star)\,(x - x^\star) \tag{C.1.1}$$

for $g(x)$. Given the algorithm is at x_n, the update step sets x_{n+1} to be the solution of $L(x_{n+1}) = 0$ where $L(x) = g(x_n) + g'(x_n)\,(x - x_n)$ is the linear approximation (C.1.1), i.e., the line whose slope is the tangent to $g(x)$ at x_n and passes through the point $(x_n, g(x_n))$. Solving $L(x_{n+1}) = 0$ gives

$$x_{n+1} = x_n - g(x_n)/g'(x_n).$$

Under conditions on the closeness of the starting value x_1 of the algorithm to a zero of $g(x)$ and on the smoothness of $g(x)$, the NR algorithm has the virtue that it converges as the square of the difference between the zero and the approximation. However, the NR method has several weaknesses as Fig. C.1 suggests. If the starting point is too far from the global minimizer, NR may converge to a local minimum. (The portion of the x region where the algorithm converges to a given local minimizer is sometimes referred to as the domain of attraction to that minimizer.) The algorithm can fail to converge. If a step produces an x_n that is a stationary point of $g(x)$, the derivative will be zero at the next update, and NR will terminate due to a division by zero.

Applied to the problem of minimizing a given $f(\cdot)$, the NR update for finding a stationary point is

$$x_{n+1} = x_n - f'(x_n)/f''(x_n).$$

When $f(\cdot)$ depends on d inputs, i.e., $\boldsymbol{x} = (x_1, \ldots, x_d)$, the update formula becomes

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - (H(\boldsymbol{x}_n))^{-1} \nabla f'(\boldsymbol{x}_n),$$

where $H(\boldsymbol{x}_n)$ is the $d \times d$ Hessian matrix whose $(i, j)^{th}$ element is $\dfrac{\partial^2 f(\boldsymbol{x})}{\partial x_i\, \partial x_j}$. Lastly, when the Hessian is difficult to compute, Quasi-Newton methods approximate it by a variety of numerical differencing methods (see, e.g., Nocedal and Wright (2006)).

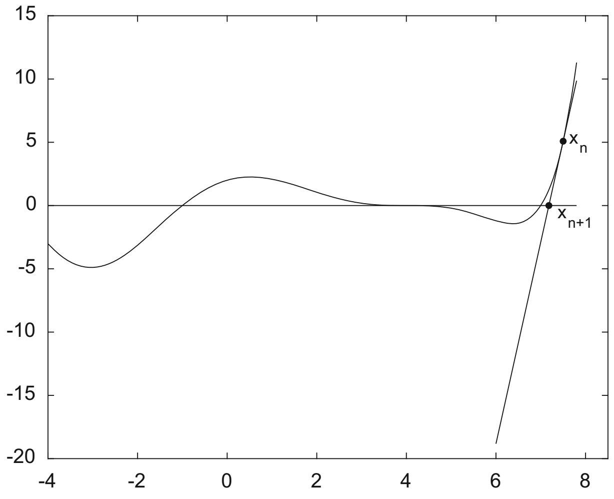


Fig. C.1 Update step of the Newton–Raphson algorithm

C.2 Direct Search Algorithms

Direct search algorithms optimize $f(\boldsymbol{x})$, $\boldsymbol{x} \in \mathcal{X}$ in cases where first derivatives are not available or are difficult to obtain. Such algorithms use only "direct" $f(\boldsymbol{x})$ evaluations to identify promising directions to update a current guess $\boldsymbol{x}_n$ of an optimal input.

C.2.1 Nelder–Mead Simplex Algorithm

Arguably the most famous direct search algorithm used in statistical applications is the Nelder–Mead simplex algorithm which was proposed to find a global minimum of a function $f(x_1, \ldots, x_d)$ of d input variables (Nelder and Mead (1965)). The method is heuristic. To describe the update method of the algorithm, suppose there are $d = 2$ input variables, say $\boldsymbol{x} = (x_1, x_2)$. The algorithm forms a triangle whose vertices consist of three input pairs. The update step compares the function values at these three vertices. The vertex with the *largest* $f(x_1, x_2)$ value is deleted and replaced with a new vertex which is selected along the line determined by the midpoint of the remaining two points and the deleted input. The new input is determined by calculating $f(\boldsymbol{x})$ at trial input values that are reflections in the opposite direction of the deleted input, the direction of apparent function descent, or, if needed, expansions or contractions along the reflection line. Thus the algorithm generates a sequence of triangles, along which the function values at the vertices become "smaller." The size of the triangles is reduced and the coordinates of the minimum point are found. For problems with d inputs, the triangles consist of simplexes having $d + 1$ input vectors. The algorithm is provably convergent for small d (Lagarias et al. (1998)) but can be slow to converge.

C.2.2 Generalized Pattern Search and Surrogate Management Framework Algorithms

Booker et al. (1999) implemented a direct search algorithm called the *Surrogate Management Framework* for the global minimization of an output $f(\boldsymbol{x})$ from an *expensive computer model* where the inputs are assumed to be located in a *rectangular* region $\mathcal{X} \subset \mathbb{R}^d$. The Surrogate Management Framework is adapted from an earlier optimization algorithm devised for *easily-computed* functions which is called the *Generalized Pattern Search* (GPS) algorithm, several of whose features are described next.

GPS algorithms use a collection of vectors that forms a *positive basis* for $\mathbb{R}^d$ to perform updates. A positive basis is a set of vectors in $\mathbb{R}^d$ such that every $\boldsymbol{x} \in \mathbb{R}^d$ can be written as a nonnegative linear combination of these vectors and no proper subset of the spanning set is also a spanning set. Unlike the (usual) basis of a vector space, the number of vectors in a positive basis for $\mathbb{R}^d$ is not unique.

The update step of the GPS algorithm requires a well-defined set of input variable locations from which the next iterate is to be selected—this is called a *mesh*. The mesh must be constructed in such a way that each element $\boldsymbol{x}_0$ of the mesh has a set of neighbors for which the collection of differences between these neighbors and $\boldsymbol{x}_0$ contains a positive basis for $\mathbb{R}^d$. This condition ensures that at least one of these difference vectors is a direction of descent at $\boldsymbol{x}_0$, provided the gradient of $f(\boldsymbol{x})$ at $\boldsymbol{x}_0$ is not equal to zero. This feature leads to desirable convergence properties of the GPS.

For $n = 0, 1, \ldots$ let M_n denote the mesh on $\mathcal{X}$ corresponding to the n^{th} update of the GPS, and let $\boldsymbol{x}_n \in M_n$ denote the input at the n^{th} iterate. If $\boldsymbol{x}_n$ is in the interior of M_n, construct a subset X_n containing $\boldsymbol{x}_n$ and a set of $2d$ inputs in M_n adjacent to $\boldsymbol{x}_n$ for which the differences between these inputs and $\boldsymbol{x}_n$ form a maximal positive basis for $\mathbb{R}^d$. If $\boldsymbol{x}_n$ is on the boundary of M_n, then one constructs a positive basis with fewer elements. By construction, M_n contains the required points to form such a positive basis for any element $\boldsymbol{x}_n$. The subsets X_n are updated as the M_n are incremented in the GPS algorithm.

Example C.1. Suppose $d = 2$, $\mathcal{X} = [0, 1]^2$, and M_0 contains $\boldsymbol{x}_0 \equiv (0.5, 0.5)$ together with a lattice of four equally spaced points on $[0, 1]^2$. If the distance between the coordinates in each dimension is ϵ, $M_0 = \{(0.5, 0.5), (0.5 \pm \epsilon, 0.5), (0.5, 0.5 \pm \epsilon)\}$. ♦

The GPS algorithm iterates between Search and Poll steps. The Search and Poll steps of the GPS algorithm are global and local in nature, respectively. If no global improvements can be found in a Search step, then local improvements are sought in the following Poll step. If necessary, the mesh is refined to facilitate convergence to a stationary point.

SEARCH: *Utilizing a finite strategy, search M_n for an $\boldsymbol{x}^t$ satisfying $f(\boldsymbol{x}^t) < f(\boldsymbol{x}_n)$. If the finite search is successful, set $M_{n+1} = M_n$, $\boldsymbol{x}_{n+1} = \boldsymbol{x}^t$, increment n, and repeat* SEARCH. *Otherwise, proceed to* POLL.

POLL: *If there exists $\boldsymbol{x}^t \in X_n$ satisfying $f(\boldsymbol{x}^t) < f(\boldsymbol{x}_n)$, set $M_{n+1} = M_n$ and $\boldsymbol{x}_{n+1} = \boldsymbol{x}^t$. If not, set $M_{n+1} = M_n/2$ (where division by two indicates halving the mesh) and $\boldsymbol{x}_{n+1} = \boldsymbol{x}_n$. Increment n and continue with* SEARCH.

Example C.1 *(Continued).* The halved mesh of M_0 is the equally spaced lattice denoted by $M_1 = M_0/2$ where the horizontal or vertical distance between two points in M_1 is $\epsilon/2$. If $M_0 = \{(0.5, 0.5), (0.5 \pm \epsilon, 0.5), (0.5, 0.5 \pm \epsilon)\}$, then M_1 is $\{(0.5, 0.5), (0.5 \pm \epsilon/2, 0.5), (0.5, 0.5 \pm \epsilon/2)\}$. ♦

Booker et al. (1999) proved that a limit point of the sequence $\{\boldsymbol{x}_n\}$ in $\mathcal{X}$ generated by GPS is a stationary point of $f(\cdot)$ provided $f(\cdot)$ is continuously differentiable on $\mathcal{X}$.

While the GPS algorithm is used to minimize *easy-to-calculate* functions, the Surrogate Management Framework (SMF) is an adaptation of GPS designed to minimize *expensive-to-calculate* (computer simulator) outputs $f(\boldsymbol{x})$. SMF algorithms use an $f(\boldsymbol{x})$ predictor. An SMF utilizing a selected EBLUP of $f(\boldsymbol{x})$ is described in the following paragraphs.

The EBLUP is first calculated from responses generated on an initial experimental design in the same way as most of the sequential algorithms described in Chap. 6. Let $\widehat{f_n}$ denote the EBLUP at iteration n. The critical feature of the SMF algorithm is the selection of a *trial set* T_n within the stage n mesh, M_n; T_n typically consists of a single point although this need not be the case. The trial set is determined using $\widehat{f_n}$, and several options are possible. One method is to construct T_n to contain a minimizer obtained from a finite-difference quasi-Newton method applied to $\widehat{f_n}$. The SMF algorithm iterates between SEARCH and POLL steps, analogous to GPS, with an additional EVALUATE/CALIBRATE step that updates the predicted response as the algorithm progresses.

SEARCH: *Choose a trial set $T_n \subset M_n$ (T_n can depend on the predictor $\widehat{f_n}$ of $f(\cdot)$). If $T_n \neq \emptyset$, it must contain at least one point at which $f(\cdot)$ is unknown. If $T_n = \emptyset$, proceed to* POLL.

EVALUATE/CALIBRATE: *Evaluate $f(\boldsymbol{x})$ for $\boldsymbol{x} \in T_n$. Stop evaluation if $\boldsymbol{x}^t \in T_n$ is found such that $f(\boldsymbol{x}^t) < f(\boldsymbol{x}_n)$. Update $\widehat{f_n}$ with the new output generated from these model runs; set $M_{n+1} = M_n$, $\boldsymbol{x}_{n+1} = \boldsymbol{x}_t$, $\widehat{f_{n+1}} = \widehat{f_n}$, increment n, and repeat* SEARCH. *If not, repeat* SEARCH *with the updated $\widehat{f_n}$ but without incrementing n.*

POLL: *Identical to the GPS* POLL. *In addition, prior to incrementing n, update $\widehat{f_n}$ to $\widehat{f_{n+1}}$ with any new model runs.*

Booker et al. (1999) show that SMF inherits the GPS convergence properties. The efficiency of SMF is determined by how effective the search component of the algorithm is implemented to find the trial set T_n.

C.2.3 DIRECT Algorithm

The `DIRECT` optimization algorithm was introduced by Jones et al. (1993) to solve (difficult) global minimization problems having a bounded hyper-rectangular input region and a real-valued objective function. `DIRECT` requires no knowledge of the objective function gradient. The name `DIRECT` comes from shortening the phrase *DIviding RECTangle*, which describes the way the algorithm moves toward the optimum. Stated formally, given $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^N$ and $\epsilon > 0$, `DIRECT` is meant to find $\boldsymbol{x}_{opt} \in \Omega$ such that

$$f(\boldsymbol{x}_{opt}) \leq \inf f(\boldsymbol{x}) + \epsilon.$$

where $\Omega = \left\{\boldsymbol{x} \in \mathbb{R}^N : a_i \leq x_i \leq b_i, 1 \leq i \leq N\right\}$.

`DIRECT` samples points from rectangular regions in the domain and uses this information to decide where to search next. The DIRECT algorithm is provably convergent to the minimizer of the objective function under smoothness conditions on $f(\boldsymbol{x})$ (Jones et al. (1993)). As for the Nelder–Mead algorithm, global convergence may come at the cost of a large and exhaustive search over the domain.

C.3 Genetic/Evolutionary Algorithms

C.3.1 Simulated Annealing

Simulated annealing is a global minimization algorithm for $f(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}$ that can be used when $f(\boldsymbol{x})$ has numerous local minima. The algorithm can also be used when $\mathcal{X}$ is discrete, as would be the case in determining a maximin design within the class of Latin hypercube designs (see Sect. 5.2). For example, Morris and Mitchell (1995) used simulated annealing to find an LHD that maximized a criterion similar to the ARD criterion defined in (5.4.5).

Unlike most of the previously sketched algorithms, each update step of simulated annealing either moves in a direction that decreases $f(\boldsymbol{x})$ or, with a specified probability, moves in a random direction that need not decrease $f(\boldsymbol{x})$, a "bad" direction. Myopically searching in directions that only decrease $f(\boldsymbol{x})$ can lead to finding only local minima. Allowing updates in randomly selected "bad" directions allows the algorithm to move away from local minima and explore new parts of the input space $\mathcal{X}$ that may contain a global minimum. After exploring many such random directions and observing that the $f(\boldsymbol{x})$ declines only slowly, simulated annealing changes its objective and limits the size of the permitted "bad" directions.

The heuristic employed by simulated annealing is derived by observing the annealing of a physical process which involves melting a solid followed by slow cooling to a minimum free energy state. During the cooling process, transitions are accepted to occur from a low- to a high-energy level through a Boltzmann probability distribution. In the final steps, the algorithm may "quench" the process by accepting only directions that decrease $f(\boldsymbol{x})$ in order to find a local minimum in the neighborhood where a global optimum is to be found (see Press et al. (2007)).

C.3.2 Particle Swarm Optimization

Particle swarm optimization (PSO) algorithms were introduced in Kennedy and Eberhart (1995) to find the global minimum of $f(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}$. The behavior of the algorithm mimics, in a sense, the behavior of a swarm of bees/ants searching for a food source. The algorithm initializes a user-specified number of starting values $\boldsymbol{x}$, called "particles," with the collection of all the particles called a "swarm." The algorithm searches $\mathcal{X}$ by updating each particle $\boldsymbol{x}_i$ for a given number of steps; at the conclusion of the steps for all particles, the algorithm selects the particle and step location for that particle which minimizes $f(\cdot)$ as the desired minimizing $\boldsymbol{x}$.

The update direction and distance of particle $\boldsymbol{x}_i$ at the current step are determined by three factors: (1) the value of $\boldsymbol{x}_i$ which produced the particle-smallest value of $f(\cdot)$ in the steps prior to the current one, (2) the value of that member of the swarm which produced the swarm-smallest value of $f(\cdot)$ in the steps prior to the current one, and (3) random movement. Thus PSO algorithms rely on randomness and "swarm communication" to identify the minimizing $\boldsymbol{x} \in \mathcal{X}$. (See Yang (2010) for an introduction to PSO.)

PSO algorithms have had many statistical applications. These include the computation of optimal designs for physical experiments using classical criteria (see Chen et al. (2014b); Phoa et al. (2015) and the references therein) and the computation of optimal designs for computer experiments (see Chen et al. (2013); Leatherman (2013)). For example, Leatherman (2013) used the output of a PSO to identify starting points for a gradient-based nonlinear optimizer (`fmincon.m` from the MATLAB Optimization toolbox) to find designs that minimized the integrated mean squared prediction error and weighted mean squared prediction error. This strategy is similar to that of Butler et al. (2014).

PSO can be easily implemented and is computationally inexpensive because its memory and CPU speed requirements are low; in addition it does not require gradient information of the objective function but only function values.

Appendix D
An Introduction to Markov Chain Monte Carlo Algorithms

This appendix describes the Gibbs and the Metropolis–Hastings (MH) algorithms for generating draws $\boldsymbol{W} = (W_1, \ldots, W_d)$ from a target density $\pi(\boldsymbol{w})$. Both are examples of Markov chain Monte Carlo (MCMC) algorithms. Readers desiring to learn additional details about MCMC algorithms can consult the following material. Casella and George (1992) provide an introductory treatment of the Gibbs algorithm, while Chib and Greenberg (1995) give an elementary description of the MH algorithm. The book length treatments of Gilks et al. (1996), Liu (2008), and Gelman et al. (2013) provide the theoretical properties of MCMC methods and discuss implementation issues.

The Gibbs algorithm assumes that all *full conditionals* of $\pi(\boldsymbol{w})$ are known and can be sampled. The full conditionals of $\pi(\boldsymbol{w})$ are the d conditional distributions $\pi(w_i \mid \boldsymbol{w}_C)$ where $C = \{1, \ldots, d\} \setminus \{i\}$, $i = 1, \ldots, d$; $\boldsymbol{w}_C$ denotes the sub-vector of $\boldsymbol{w}$ having elements w_j, $j \in C$.

Given integer $M > 0$, the Gibbs algorithm draws $\boldsymbol{w}^1, \ldots, \boldsymbol{w}^M$ as follows.

[*Initialization*] Randomly choose $\boldsymbol{w}^0$ in the support of $\pi(\boldsymbol{w})$, set $k = 1$, and go to Step 1.

[*Step 1*] Draw w_1^k from $\pi(w_1 \mid \boldsymbol{w}_C^{k-1})$, where $C = \{2, \ldots, d\}$.

[*Step 2*] Draw w_2^k from $\pi(w_2 \mid w_1^k, \boldsymbol{w}_C^{k-1})$, where $C = \{3, \ldots, d\}$.

[*Step 3*] Draw w_3^k from $\pi(w_3 \mid w_1^k, w_2^k, \boldsymbol{w}_C^{k-1})$, where $C = \{4, \ldots, d\}$.

$\vdots$

[*Step d*] Draw w_d^k from $\pi(w_d \mid w_1^k, w_2^k, \ldots, w_{d-1}^k)$. If $k = M$ go to [*Return*]; otherwise increment k to $k + 1$, and go to [*Step 1*].

[*Return*] $\boldsymbol{w}^1, \ldots, \boldsymbol{w}^M$.

Under mild conditions, $\pi(\boldsymbol{w})$ is the limiting *distribution* of the sequence draws as $M \to \infty$ (see Roberts and Smith (1994)).

T. J. Santner et al., *The Design and Analysis of Computer Experiments*, Springer Series in Statistics, https://doi.org/10.1007/978-1-4939-8847-1

Example 4.2 *(Continued)*. In this example from Chap. 4, there are test and training data that follow a two-stage model with first-stage $[(\boldsymbol{Y}^{te}, \boldsymbol{Y}^{tr}) \mid (\boldsymbol{\beta}, \lambda_Z)]$ distribution given by (4.1.1) and second stage (prior)

$$[\boldsymbol{\beta} \mid \lambda_Z] \sim N_p\left(\boldsymbol{b}_\beta, \lambda_Z^{-1}\boldsymbol{V}_\beta\right) \quad \text{and} \quad [\lambda_Z] \sim \Gamma(c, d)\,.$$

The parameter $(\boldsymbol{\beta}, \lambda_Z)$ is unknown, and the correlation parameters are known. The Gibbs algorithm for sampling from the posterior $[\boldsymbol{\beta}, \lambda_Z \mid \boldsymbol{y}^{tr}]$ is as follows.

The calculation in Sect. 4.1 uses fixed λ_Z and is thus conditional. It shows that

$$\left[\boldsymbol{\beta} \;\middle|\; \boldsymbol{Y}^{tr}, \lambda_Z\right] \sim N_p\left(\boldsymbol{\mu}_{\beta|tr}, \boldsymbol{\Sigma}_{\beta|tr}\right), \tag{D.0.1}$$

where

$$\begin{aligned}\boldsymbol{\mu}_{\beta|tr} &= \left(\lambda_Z\, \boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr} + \lambda_Z\, \boldsymbol{V}_\beta^{-1}\right)^{-1} \times \left(\lambda_Z\, \boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}} + \lambda_Z\, \boldsymbol{V}_\beta^{-1}\boldsymbol{b}_\beta\right)\\ &= \left(\boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr} + \boldsymbol{V}_\beta^{-1}\right)^{-1} \times \left(\boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr}\widehat{\boldsymbol{\beta}} + \boldsymbol{V}_\beta^{-1}\boldsymbol{b}_\beta\right)\end{aligned}$$

is independent of λ_Z and

$$\boldsymbol{\Sigma}_{\beta|tr} = \boldsymbol{\Sigma}_{\beta|tr}(\lambda_Z) = \lambda_Z^{-1}\left(\boldsymbol{F}_{tr}^\top \boldsymbol{R}_{tr}^{-1}\boldsymbol{F}_{tr} + \boldsymbol{V}_\beta^{-1}\right)^{-1}.$$

The kernel of the $[\lambda_Z \mid \boldsymbol{Y}^{tr}, \boldsymbol{\beta}]$ conditional can be written as

$$\begin{aligned}\left[\lambda_Z \;\middle|\; \boldsymbol{Y}^{tr}, \boldsymbol{\beta}\right] &\propto \left[\boldsymbol{Y}^{tr} \;\middle|\; \boldsymbol{\beta}, \lambda_Z\right][\boldsymbol{\beta} \mid \lambda_Z][\lambda_Z]\\ &\propto \lambda_Z^{c+(n_s+p)/2-1}\ \exp\left\{-\frac{\lambda_Z}{2}\,(2d + Q_1 + Q_2)\right\}\end{aligned} \tag{D.0.2}$$

where

$$Q_1 = Q_1(\boldsymbol{\beta}) = \left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\boldsymbol{\beta}\right)^\top \boldsymbol{R}_{tr}^{-1}\left(\boldsymbol{y}^{tr} - \boldsymbol{F}_{tr}\boldsymbol{\beta}\right)$$

and

$$Q_2 = Q_2(\boldsymbol{\beta}) = \left(\boldsymbol{\beta} - \boldsymbol{b}_\beta\right)^\top \boldsymbol{V}_\beta^{-1}\left(\boldsymbol{\beta} - \boldsymbol{b}_\beta\right)$$

(see Proof of Theorem 4.2 in Sect. 4.4.1). This shows

$$\left[\lambda_Z \;\middle|\; \boldsymbol{Y}^{tr}, \boldsymbol{\beta}\right] \sim \Gamma\left(c + (n_s + p)/2, d + \frac{Q_1 + Q_2}{2}\right).$$

Thus the Gibbs algorithm first selects $\lambda_Z^0 > 0$ and $\boldsymbol{\beta}^0 \in \mathbb{R}^p$, sets $k = 1$, and specifies a total number, M, of $(\boldsymbol{\beta}, \lambda_Z)$ draws. Each $\boldsymbol{\beta}^k$ is updated as a draw from

$$N_p\left(\boldsymbol{\mu}_{\beta|tr}, \boldsymbol{\Sigma}_{\beta|tr}(\lambda_Z^{k-1})\right),$$

and λ_Z^k is updated as a draw from

$$\Gamma\left(c + (n_s + p)/2, d + \frac{Q_1(\boldsymbol{\beta}^k) + Q_2(\boldsymbol{\beta}^k)}{2}\right).$$

This produces the sequence $\{\boldsymbol{\beta}^k, \lambda_Z^k\}$, $k = 1, \ldots, M$, whose limit (as $M \to \infty$) is a draw from the joint conditional $[\boldsymbol{\beta}, \lambda_Z \mid \boldsymbol{y}^{tr}]$. ♦

In contrast to the Gibbs algorithm which requires that all full conditionals of the $\boldsymbol{W}$ distribution be known, the MH algorithm requires only that the $\boldsymbol{W}$ density be known up to a constant of proportionality, i.e., that $\boldsymbol{W}$ have density $c \times \pi(\boldsymbol{w})$ where $\pi(\boldsymbol{w})$ is known but $c > 0$ need not be known. To simplify notation, the description below assumes $c = 1$, i.e., $\pi(\boldsymbol{w})$ is the target density. However, the use of $\pi(\boldsymbol{w})$ only as a ratio in Step 2 of the MH algorithm (below) makes clear that the method does not require knowledge of normalizing constants.

In overview, the MH algorithm forms the entire $\boldsymbol{W}^{k+1}$ draw at once. In contrast, the Gibbs algorithm made $\boldsymbol{W}^{k+1}$ draws one subcomponent at a time. Users must specify two quantities to implement the MH algorithm: the *number of iterations*, M, that the algorithm will run and a *proposal distribution*. Each iteration of the algorithm consists of three steps and returns the next $\boldsymbol{w}$ value. Thus at the conclusion of the algorithm a sequence of values $\boldsymbol{w}^1, \ldots, \boldsymbol{w}^M$ is determined. The second quantity that must be specified is the proposal distribution which will be denoted $q(\cdot \mid \cdot)$. Suppose $\boldsymbol{w}^k$ has been constructed at the conclusion of iteration k, $q(\boldsymbol{w}^k \mid \cdot)$ is a probability distribution used to generate a "candidate" value for $\boldsymbol{w}^{k+1}$, say $\boldsymbol{w}^{cand}$. The $q(\boldsymbol{w}^k \mid \cdot)$ notation may seem strange to readers who are more familiar with the notation $f(\cdot \mid \text{parameters})$ for a family of probability model densities *given* specific parameter values. Here $\boldsymbol{w}^k$ in $q(\boldsymbol{w}^k \mid \cdot)$ plays the role of the conditioning quantity. This notation is used in the MCMC literature because $q(\boldsymbol{w}^k \mid \cdot)$ is the continuous version of a transition probability matrix for a finite Markov chain in which one moves from state i to a new state j with a probability $P_{i,j}$. Then, at the conclusion of iteration $k + 1$, $\boldsymbol{w}^{k+1}$ is either set equal to the candidate, $\boldsymbol{w}^{cand}$, or remains at value $\boldsymbol{w}^k$. Given M and $q(\cdot \mid \cdot)$, the MH algorithm is described as follows.

[*Initialization*] Randomly choose $\boldsymbol{w}^1$ in the support of $\pi(\boldsymbol{w})$, set $k = 1$, and go to Step 1.

[*Step 1*] Given the current value $\boldsymbol{W}^k = \boldsymbol{w}^k$, let $\boldsymbol{w}^{cand}$ be the value of a draw from $q(\boldsymbol{w}^k \mid \cdot)$ and $U = u$ a draw from an independent $U(0, 1)$ random variable.

[*Step 2*] Set

$$\alpha = \alpha(\boldsymbol{w}^k, \boldsymbol{w}^{cand}) =$$

$$\begin{cases} \min\left\{ \dfrac{\pi(\boldsymbol{w}^{cand})\, q(\boldsymbol{w}^{cand} \mid \boldsymbol{w}^k)}{\pi(\boldsymbol{w}^k)\, q(\boldsymbol{w}^k \mid \boldsymbol{w}^{cand})}, 1 \right\}, & \pi(\boldsymbol{w}^k)\, q(\boldsymbol{w}^k \mid \boldsymbol{w}^{cand}) > 0 \\ 1, & \text{o.w.} \end{cases} .$$

[*Step 3*] Set

$$\boldsymbol{w}^{k+1} = \begin{cases} \boldsymbol{w}^{cand}, & \text{if} \quad u \le \alpha(\boldsymbol{w}^k, \boldsymbol{w}^{cand}) \\ \boldsymbol{w}^k, & \text{if} \quad u > \alpha(\boldsymbol{w}^k, \boldsymbol{w}^{cand}) \end{cases} .$$

If $k = M$ go to [*Return*], otherwise increment k to $k + 1$ and go to [*Step 1*].

[*Return*] $\boldsymbol{w}^1, \ldots, \boldsymbol{w}^M$

As is true for the Gibbs algorithm, $\pi(\boldsymbol{w})$ is the limiting *distribution* of the $\boldsymbol{W}$ sequence as $M \to \infty$ under mild conditions (see Roberts and Smith (1994)).

In practice, both Gibbs and MH algorithms discard an initial number of "burn-in" draws $\boldsymbol{w}^1, \ldots, \boldsymbol{w}^M$ and treat the remainder as dependent draws from $\pi(\boldsymbol{w})$. In the examples of Chap. 4, a total of $M = 10{,}000$ draws were made of which the first 5000 were discarded as burn-in.

The choice of the transition kernel $q(\cdot \mid \cdot)$ is crucial to the success of the algorithm. In Chaps. 4 and 8, $q(\boldsymbol{w}^k \mid \cdot)$ was taken to be uniformly distributed about the center point $\boldsymbol{w}^k$ with a (support) width that is determined to provide a "good" acceptance rate in Step 3. The uniform proposal distribution has the property that $q(\boldsymbol{w}^{cand} \mid \boldsymbol{w}^k) = q(\boldsymbol{w}^k \mid \boldsymbol{w}^{cand})$ so that $\alpha(\boldsymbol{w}^k, \boldsymbol{w}^{cand})$ is independent of $q(\cdot \mid \cdot)$. If the width of the proposal distribution is "too small," the proposals will be accepted with high probability, but the $\boldsymbol{w}^k$ sequence will move only slightly from iteration to iteration; a (very) large M may be required to achieve the $\pi(\boldsymbol{w})$ limiting behavior. If the width is "too large," the proposal can be "far" from the current $\boldsymbol{w}^k$ and often be rejected; the result is a $\boldsymbol{w}^k$ sequence that is stuck on a few points. Typically, one must "train" the MH algorithm by selecting the widths of the proposal distribution to achieve a target acceptance rate. Gelman et al. (1996) show that in certain theoretical circumstances, a rate of about 44% is optimal for a univariate $\boldsymbol{w}$. Graves (2011) and Roberts and Rosenthal (2007) present two different approaches for training an MH algorithm.

We conclude by noting that the Gibbs and MH algorithms can be blended. As an example, assume that the correlation parameters $\boldsymbol{\rho}$ in the regression + stationary GP model in Example 4.2 are *unknown*. It is desired to generate draws from $[(\boldsymbol{\beta}, \lambda_Z, \boldsymbol{\rho}) \mid \boldsymbol{Y}^{tr}]$. Each iteration of the Gibbs algorithm requires draws from the three full conditionals $[\boldsymbol{\beta} \mid \boldsymbol{Y}^{tr}, \lambda_Z, \boldsymbol{\rho}]$, $[\lambda_Z \mid \boldsymbol{Y}^{tr}, \boldsymbol{\beta}, \boldsymbol{\rho}]$, and $[\boldsymbol{\rho} \mid \boldsymbol{Y}^{tr}, \boldsymbol{\beta}, \lambda_Z]$. The calculations in Eqs. (D.0.1) and (D.0.2) with known $\boldsymbol{\rho}$ provide the full conditionals $[\boldsymbol{\beta} \mid \boldsymbol{Y}^{tr}, \lambda_Z, \boldsymbol{\rho}]$ and $[\lambda_Z \mid \boldsymbol{Y}^{tr}, \boldsymbol{\beta}, \boldsymbol{\rho}]$, respectively. It is straightforward to generate draws from each. The conditional $[\boldsymbol{\rho} \mid \boldsymbol{Y}^{tr}, \boldsymbol{\beta}, \lambda_Z]$ does not have a recognizable form. However,

$$\left[\boldsymbol{\rho} \;\middle|\; \boldsymbol{Y}^{tr}, \boldsymbol{\beta}, \lambda_Z\right] \propto \left[\boldsymbol{Y}^{tr} \;\middle|\; \boldsymbol{\beta}, \lambda_Z, \boldsymbol{\rho}\right] \times [\boldsymbol{\rho}]$$

so that the desired conditional distribution is known up a normalizing constant. The MH algorithm can be used to draw an observation from this distribution and provides the third step of a Gibbs algorithm for sampling from $[(\boldsymbol{\beta}, \lambda_Z, \boldsymbol{\rho}) \mid \boldsymbol{Y}^{tr}]$.

Appendix E
A Primer on Constructing Quasi-Monte Carlo Sequences

To construct a Halton sequence $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ of n points on the d-dimensional unit hypercube, begin by choosing d prime numbers or bases, $b_1, b_2, \ldots, b_d$. These could be, for example, the first d prime numbers. The base b_j is used to construct the j^{th} coordinate of each of $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$, $j = 1, \ldots, d$.

Next, select an integer m. Fix $j \in \{1, \ldots, d\}$. Represent the integer m in base b_j as

$$m = \sum_{k=0}^{t_{mj}} a_{jk}(m) b_j^k \quad \text{for} \quad j = 1, \ldots, d. \tag{E.0.1}$$

where each $a_{jk}(m) \in \{0, 1, \ldots, (b_j - 1)\}$ and t_{mj} is the highest power of b_j used in the representation of m in base b_j. Form $x_{1,j}$ by reversing the digits in the representation of m in base b_j and placing these reversed digits after a decimal point, i.e.,

$$x_{1,j} = \sum_{k=0}^{t_{mj}} a_{j,t_{mj}-k}(m) b_j^{k-t_{mj}-1} \quad \text{for} \quad j = 1, \ldots, d. \tag{E.0.2}$$

Set $m = m + i - 1$ and repeat the above to form $\boldsymbol{x}_i = (x_{i,1}, \ldots, x_{i,d})$ for $i = 2, \ldots, n$.

Example E.1 (Constructing a Halton Sequence). We compute the first 5 (= n) points in a two-dimensional Halton sequence ($d = 2$) corresponding to the bases $b_1 = 2$ and $b_2 = 3$. We take $m = 4$. The representation (E.0.1) of 4 in base 2 is 100 ($j = 1$) and is 11 in base 3 ($j = 2$); here $t_{m1} = 2$ and $t_{m2} = 1$. Reversing the digits and adding a decimal point give $\boldsymbol{x}_1 = (.001_2, .11_3)$ for Eq. (E.0.2) where the subscript indicates the base. Converting to base 10,

$$.001_2 = 0 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} = 1/8 = 0.125$$

and

$$.11_3 = 1 \times 3^{-1} + 1 \times 3^{-2} = 1/3 + 1/9 = 0.444.$$

Thus, the first point in the Halton sequence is $\boldsymbol{x}_1 = (.125, .444)$.

To compute $\boldsymbol{x}_2$, increase m by 1 to the value 5 (setting $i = 2$). The value 5 is 101 in base 2 and is 12 in base 3. Reversing the digits and adding a decimal point, $\boldsymbol{x}_2 = (.101_2, .21_3)$. Converting to base 10,

$$.101_2 = 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} = 1/2 + 1/8 = 0.625$$

and

$$.21_3 = 2 \times 3^{-1} + 1 \times 3^{-2} = 2/3 + 1/9 = 0.7784.$$

Thus, the second point in our Halton sequence is $\boldsymbol{x}_2 = (.625, .778)$.

The next three points correspond to $m = 6$, 7, and 8. In base 2, these m are 110, 111, and 1000, respectively. In base 3, these are 20, 21, and 22. Reversing digits and adding a decimal point gives $\boldsymbol{x}_3 = (.011_2, .02_3)$, $\boldsymbol{x}_4 = (.111_2, .12_3)$, and $\boldsymbol{x}_5 = (.0001_2, .22_3)$. Converting to base 10, one finds $\boldsymbol{x}_3 = (.375, .222)$, $\boldsymbol{x}_4 = (.875, .556)$, and $\boldsymbol{x}_5 = (.0625, .8889)$. Figure E.1 shows the resulting five-point design. ♦

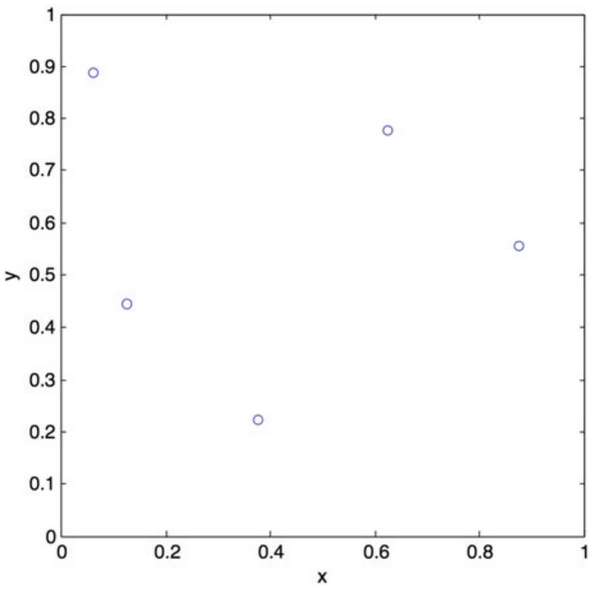


Fig. E.1 An $n = 5$ point, $d = 2$ variable Halton sequence

Halton sequences are relatively easy to calculate and have been found to be acceptably uniform for lower dimensions (d up to about 10). For higher dimensions the quality degrades rapidly because two-dimensional planes occur in cycles with decreasing periods. Methods for creating sequences that behave better (appear uniform even in higher dimensions) have been developed by Sobol´ and further improved by Niederreiter (Sobol´ (1967), Sobol´ (1976), Niederreiter (1988), and Lemieux (2009)).

References

Abrahamsen P (1997) A review of Gaussian random fields and correlation functions. Technical report 917, Norwegian Computing Center, Box 114, Blindern, N0314 Oslo

Adler RJ (1981) The geometry of random fields. Wiley, New York, NY

Adler RJ (1990) An introduction to continuity, extrema, and related topics for general Gaussian processes. Institute of Mathematical Statistics, Hayward, CA

Allen DM (1974) The relationship between variable selection and data augmentation and a method for prediction. Technometrics 16:125–127

Andrieu C, Thoms J (2008) A tutorial on adaptive MCMC. Stat Comput 18(4): 343–373

Atamturktur S, Williams B, Egeberg M, Unal C (2013) Batch sequential design of optimal experiments for improved predictive maturity in physics-based modeling. Struct Multidiscip Optim 48:549–569

Atamturktur S, Hegenderfer J, Williams B, Unal C (2015) Selection criterion based on an exploration-exploitation approach for optimal design of experiments. ASCE: J Eng Mech 141(1):04014108

Atkinson AC, Donev AN (1992) Optimum experimental designs. Clarendon Press, Oxford

Ba S, Joseph VR (2011) Multi-layer designs for computer experiments. J Am Stat Assoc 106:1139–1149

Ba S, Joseph VR (2012) Composite Gaussian Process models for emulating expensive functions. Ann Appl Stat 6(4):1838–1860

Ba S, Myers WR, Brenneman WA (2015) Optimal sliced Latin hypercube designs. Technometrics 57(4):479–487

Bachoc F (2013) Cross validation and maximum likelihood estimations of hyper-parameters of Gaussian processes with model misspecification. Comput Stat Data Anal 66:55–69

Balling R (2003) The maximin fitness function: A multiobjective city and regional planning. In: Fonseca C, Fleming P, Zitzler E, Deb K, Thiele L (eds) Evolutionary multi-criterion optimization. Springer, Berlin, pp 1–15

Banerjee S, Gelfand AE, Finley AO, Sang H (2008) Gaussian predictive process models for large spatial data sets. J R Stat Soc Ser B 70(4):825–848

Barton RR (1999) Graphical methods for the design of experiments. Lecture notes in statistics, vol 143. Springer, New York, NY

Bastos LS, O'Hagan A (2009) Diagnostics for Gaussian process emulators. Technometrics 51(4):425–438

Bates RA, Buck RJ, Riccomagno E, Wynn HP (1996) Experimental design and observation for large systems. J R Stat Soc Ser B 58:77–94

Bautista DC (2009) A sequential design for approximating the Pareto front using the expected Pareto improvement function. PhD thesis, Department of Statistics, The Ohio State University, Columbus, OH

Bayarri MJ, Berger JO, Paulo R, Sacks J, Cafeo JA, Cavendish J, Lin CH, Tu J (2007) A framework for validation of computer models. Technometrics 49(2):138–154

Beattie SD, Lin DK (1997) Rotated factorial design for computer experiments. In: ASA proceedings of the section on physical and engineering sciences, American Statistical Association, Alexandria, VA, pp 431–450

Ben-Ari EN, Steinberg DM (2007) Experiments: an empirical comparison of kriging with MARS and projection pursuit regression. Qual Eng 19:327–338

Berger JO (1985) Statistical decision theory and Bayesian analysis. Springer, New York, NY

Berger JO, De Oliveira V, Sansó B (2001) Objective Bayesian analysis of spatially correlated data. J Am Stat Assoc 96:1361–1374

Berk R, Bickel P, Campbell K, Fovell R, Keller-McNulty S, Kelly E, Linn R, Park B, Perelson A, Rouphail N, Sacks J, Schoenberg F (2002) Workshop on statistical approaches for the evaluation of complex computer models. Stat Sci 17(2):173–192

Bernardo MC, Buck RJ, Liu L, Nazaret WA, Sacks J, Welch WJ (1992) Integrated circuit design optimization using a sequential strategy. IEEE Trans Comput Aided Des 11:361–372

Birk DM (1997) An introduction to mathematical fire modeling. Technomic Publishing, Lancaster

Blatman G, Sudret B (2010) Efficient computation of global sensitivity indices using sparse polynomial chaos expansions. Reliab Eng Syst Saf 95:1216–1229

Bliznyuk N, Ruppert D, Shoemaker C, Regis R, Wild S, Mugunthan P (2008) Bayesian calibration and uncertainty analysis for computationally expensive models using optimization and radial basis function approximation. J Comput Graph Stat 17:270–294

Bochner S (1955) Harmonic analysis and the theory of probability. University of California Press, Berkeley, CA

Booker AJ, Dennis JE, Frank PD, Serafini DB, Torczon V (1997) Optimization using surrogate objectives on a helicopter test example. Technical report SSGTECH-97-027, Boeing technical report

Booker AJ, Dennis JE, Frank PD, Serafini DB, Torczon V, Trosset MW (1999) A rigorous framework for optimization of expensive functions by surrogates. Struct Optim 17:1–13

Bornn L, Shaddick G, Zidek JV (2012) Modeling nonstationary processes through dimension expansion. J Am Stat Assoc 107:281–289

Box GE, Draper NR (1987) Empirical model-building and response surfaces. Wiley, New York, NY

Box G, Hunter W, Hunter J (1978) Statistics for experimenters. Wiley, New York, NY

Bratley P, Fox BL, Niederreiter H (1994) Algorithm 738: programs to generate Niederreiter's low-discrepancy sequences. ACM Trans Math Softw 20:494–495

Brynjarsdóttir J, O'Hagan A (2014) Learning about physical parameters: the importance of model discrepancy. Inverse Prob 30:114007 (24pp)

Buhmann MD (2003) Radial basis functions: theory and implementations. Cambridge University Press, Cambridge

Bursztyn D, Steinberg DM (2002) Rotation designs: orthogonal first-order designs with higher-order projectivity. J Appl Stoch Models Bus Ind 18:197–206

Bursztyn D, Steinberg DM (2006) Comparison of designs for computer experiments. J Stat Plann Inf 136:1103–1119

Butler NA (2001) Optimal and orthogonal Latin hypercube designs for computer experiments. Biometrika 88:847–857

Butler A, Haynes RD, Humphries TD, Ranjan P (2014) Efficient optimization of the likelihood function in Gaussian process modelling. Comput Stat Data Anal 73:40–52

Campbell K (2001) Functional sensitivity analysis for computer model output. In: Bodt BA, Wegman EJ (eds) Proceedings of the seventh U.S. army conference on applied statistics. Army Research Laboratory, pp 35–46

Campbell KS, McKay MD, Williams BJ (2005) Sensitivity analysis when model outputs are functions (tutorial). In: Hanson KM, Hemez FM (eds) Proceedings of the SAMO 2004 conference on sensitivity analysis. Los Alamos National Laboratory, Los Alamos, NM, pp 81–89. https://library.lanl.gov/cgi-bin/getdoc?event=SAMO2004&document=samo04-proceedings.pdf

Campbell KS, McKay MD, Williams BJ (2006) Sensitivity analysis when model outputs are functions. Reliab Eng Syst Saf 91:1468–472

Campolongo F, Cariboni J, Saltelli A (2007) An effective screening design for sensitivity analysis of large models. Environ Model Softw 22:1509–1518

Campolongo F, Saltelli A, Cariboni J (2011) From screening to quantitative sensitivity analysis. A unified approach. Comput Phys Commun 43:39–52

Casella G, George E (1992) Explaining the Gibbs sampler. Am Stat 46:167–174

Chakraborty A, Mallick BK, McClarren RG, Kuranz CC, Bingham D, Grosskopf MJ, Rutter EM, Stripline HF, Drake RP (2013) Spline-based emulators for radiative shock experiments with measurement error. J Am Stat Assoc 108:411–428

Chakraborty A, Bingham D, Dhavala SS, Kuranz CC, Drake RP, Grosskopf MJ, Rutter EM, Torralva BR, Holloway JP, McClaren RG, Malllick BK (2017) Emulation of numerical models with over-specified basis functions. Technometrics 59:153–164

Chang PB (1998) Robust design and analysis of femoral components for total hip arthroplasty. PhD thesis, Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY

Chang PB, Williams BJ, Bawa Bhalla KS, Belknap TW, Santner TJ, Notz WI, Bartel DL (2001) Robust design and analysis of total joint replacements: finite element model experiments with environmental variables. J Biomech Eng 123:239–246

Chapman WL, Welch WJ, Bowman KP, Sacks J, Walsh JE (1994) Arctic sea ice variability: model sensitivities and a multidecadal simulation. J Geophys Res 99(C1):919–935

Chen W, Jin R, Sudjianto A (2005) Analytical variance-based global sensitivity analysis in simulation-based design under uncertainty. J Mech Des 127:875–876

Chen W, Jin R, Sudjianto A (2006) Analytical global sensitivity analysis and uncertainty propogation for robust design. J Qual Technol 38:333–348

Chen RB, Wang W, Wu CFJ (2011) Building surrogates with overcomplete bases in computer experiments with applications to bistable laser diodes. IEE Trans 182:978–988

Chen RB, Hsieh DN, Hung Y, Wang W (2013) Optimizing Latin hypercube designs by particle swarm. Stat Comput 23(5):663–676

Chen PH, Dean A, Santner T (2014a) Multivariate Gaussian process interpolators with varying-parameter covariance—with an application to Pareto front estimation. Poster Presentation, 2014 meeting of the American Statistical Association

Chen RB, Chang SP, Wang W, Tung HC, Wong WK (2014b) Minimax optimal designs via particle swarm optimization methods. Stat Comput 24:1063–1080

Chib S, Greenberg E (1995) Understanding the Metropolis-Hastings algorithm. Am Stat 49(4):327–335

Chipman H, George E, McCulloch R (1998) Bayesian CART model search (with discussion). J Am Stat Assoc 93:935–960

Chipman H, George E, McCulloch R (2002) Bayesian treed models. Mach Learn 48:303–324

Christiansen CL, Morris CN (1997) Hierarchical Poisson regression modeling. J Am Stat Assoc 92:618–632

Cioppa TM, Lucas TW (2007) Efficient nearly orthogonal and space-filling Latin hypercubes. Technometrics 49:45–55

Coello Coello CA, Lamont GB, Van Veldhuizen DA (2006) Evolutionary algorithms for solving multi-objective problems (genetic and evolutionary computation). Springer, New York, NY

Constantine PG, Dow E, Wang Q (2014) Active subspace methods in theory and practice: applications to kriging surfaces. SIAM J Sci Comput 36(4):A1500–A1524

Conti S, O'Hagan A (2010) Bayesian emulation of complex multi-output and dynamic computer models. J Stat Plann Inf 140(3):640–651

Cooper LY (1980) Estimating safe available egress time from fires. Technical report 80-2172, National Bureau of Standards, Washington

Cooper LY (1997) VENTCF2: an algorithm and associated FORTRAN 77 subroutine for calculating flow through a horizontal ceiling/floor vent in a zone-type compartmental fire model. Fire Safe J 28:253–287

Cooper LY, Stroup DW (1985) ASET—a computer program for calculating available safe egress time. Fire Safe J 9:29–45

Cox DD, Park JS, Singer CE (2001) A statistical method for tuning a computer code to a data base. Comput Stat Data Anal 37(1):77–92

Craig PC, Goldstein M, Rougier JC, Seheult AH (2001) Bayesian forecasting for complex systems using computer simulators. J Am Stat Assoc 96:717–729

Cramér H, Leadbetter MR (1967) Stationary and related stochastic processes. Wiley, New York, NY

Cressie N, Wikle CK (2011) Statistics for spatio-temporal data. Wiley, New York, NY

Cressie NA (1993) Statistics for spatial data. Wiley, New York, NY

Crick MJ, Hofer E, Jones JA, Haywood SM (1988) Uncertainty analysis of the food chain and atmospheric dispersion modules of MARC. Technical report NRPBR184, National Radiological Protection Board

Currin C, Mitchell TJ, Morris MD, Ylvisaker D (1991) Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments. J Am Stat Assoc 86:953–963

Dandekar R, Kirkendall N (1993) Latin hypercube sampling for sensitivity and uncertainty analysis. In: American Statistical Association proceedings of the Section on Physical and Engineering Sciences. American Statistical Association, Alexandria, VA, pp 26–31

Davis C (2015) A Bayesian approach to prediction and variable selection using nonstationary Gaussian processes. PhD thesis, Department of Statistics, The Ohio State University, Columbus, OH

Dean AM, Voss D, Draguljic D (2017) Design and analysis of experiments, 2nd edn. Springer, New York, NY

Dette H, Pepelyshev A (2010) Generalized Latin hypercube designs for computer experiments. Technometrics 52:421–429

Dixon LCW, Szego GP (1978) The global optimisation problem: an introduction. In: Dixon LCW, Szego GP (eds) Towards global optimisation, vol 2. North Holland, Amsterdam, pp 1–15

Doob JL (1953) Stochastic processes. Wiley, New York, NY

Draguljić D, Santner TJ, Dean AM (2012) Non-collapsing spacing-filling designs for bounded polygonal regions. Technometrics 54:169–178

Draper NR, Smith H (1981) Applied regression analysis. Wiley, New York, NY

Dunnett CW (1989) Multivariate normal probability integrals with product correlation structure (correction: 42:709). Appl Stat 38:564–579

Emmerich MT, Giannakoglou KC, Naujoks B (2006) Single- and multiobjective evolutionary optimization assisted by Gaussian random field metamodels. IEEE Trans Evol Comput 10(4):421–439

Fang KT, Lin DKJ, Winker P, Zhang Y (2000) Uniform design: theory and application. Technometrics 42:237–248

Fang KT, Li R, Sudjianto A (2006) Design and modeling for computer experiments. Chapman & Hall/CRC, Boca Raton, FL

Fonseca CM, Paquete L, López-Ibáñez M (2006) An improved dimension-sweep algorithm for the hypervolume indicator. In: IEEE congress on evolutionary computation. IEEE Press, New York, NY, pp 1157–1163

Forrester A, Sóbester A, Keane A (2007) Multi-fidelity optimization via surrogate modeling. Proc R Soc A 463(2088):3251–3269

Francom D, Sansó B, Kupresanin A, Johannesson G (2018) Sensitivity analysis and emulation for functional data using Bayesian adaptive splines. Stat Sinica 28(2):791–816

Franey M, Ranjan P, Chipman H (2011) Branch and bound algorithms for maximizing expected improvement functions. J Stat Plann Inf 141:42–55

Fricker TE, Oakley JE, Urban NM (2013) Multivariate Gaussian process emulators with nonseparable covariance structures. Technometrics 55:47–56

Gelfand AE, Schmidt AM, Banerjee S, Sirmans CF (2004) Nonstationary multivariate process modeling through spatially varying coregionalization (with discussion). Test 13:263–312

Gelman A, Roberts G, Gilks W (1996) Efficient Metropolis jumping rules. In: Bernardo J, Berger J, Dawid A, Smith A (eds) Bayesian statistics 5, vol 5, Oxford University Press, Oxford, pp 599–607

Gelman A, Carlin J, Stern H, Dunson D, Vehtari A, Rubin D (2013) Bayesian data analysis. Chapman & Hall/CRC, Boca Raton, FL

Gibbs MN (1997) Bayesian Gaussian processes for regression and classification. PhD thesis, Cambridge University, Cambridge

Gilks WR, Richardson S, Spiegelhalter D (eds) (1996) Markov chain Monte Carlo in practice. Chapman & Hall/CRC, Boca Raton, FL

Gneiting T (2002) Compactly supported correlation functions. J Multivar Anal 83(2):493–508

Gneiting T, Kleiber W, Schlather M (2010) Matérn cross-covariance functions for multivariate random fields. J Am Stat Assoc 105:1167–1177

Golub GH, Heath M, Wahba G (1979) Generalized cross-validation as a method for choosing a good ridge parameter. Technometrics 21:215–223

Gramacy RB, Lee HKH (2008) Bayesian treed Gaussian process models with an application to computer modeling. J Am Stat Assoc 103:1119–1130

Gramacy RB, Gray GA, Le Digabel S, Lee HKH, Ranjan P, Wells G, Wild SM (2016) Modeling an augmented Lagrangian for blackbox constrained optimization (with discussion). Technometrics 58(1):1–29

Graves TL (2011) Automatic step size selection in random walk Metropolis algorithms. Technical report LA-UR-11-01936, Los Alamos National Laboratory, Los Alamos, NM

Guttorp P, Sampson PD (1994) Methods for estimating heterogeneous spatial covariance functions with environmental applications. In: Patil GP, Rao CR (eds) Handbook of statistics, vol 12. Elsevier Science B.V., Amsterdam, pp 661–689

Guttorp P, Meiring W, Sampson PD (1994) A space-time analysis of ground-level ozone data. Environmetrics 5:241–254

Haario H, Laine M, Mira A, Saksman E (2006) DRAM: efficient adaptive MCMC. Stat Comput 16(4):339–354

Haas TC (1995) Local prediction of a spatio-temporal process with an application to wet sulfate deposition. J Am Stat Assoc 90:1189–1199

Halton JH (1960) On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. Numer Math 2(1):84–90

Han G, Santner TJ, Notz WI, Bartel DL (2009a) Prediction for computer experiments having quantitative and qualitative input variables. Technometrics 51(2):278–288

Han G, Santner TJ, Rawlinson JJ (2009b) Simultaneous determination of tuning and calibration parameters for computer experiments. Technometrics 51(4):464–474

Handcock MS (1991) On cascading Latin hypercube designs and additive models for experiments. Commun Stat Theory Methods 20(2):417–439

Handcock MS, Stein ML (1993) A Bayesian analysis of kriging. Technometrics 35:403–410

Handcock MS, Wallis JR (1994) An approach to statistical spatial-temporal modeling of meterological fields. J Am Stat Assoc 89:368–390

Harper WV, Gupta SK (1983) Sensitivity/uncertainty analysis of a borehole scenario comparing Latin hypercube sampling and deterministic sensitivity approaches. Technical report BMI/ONWI-516, Office of Nuclear Waste Isolation, Battelle Memorial Institute, Columbus, OH

Harville DA (1974) Bayesian inference for variance components using only error contrasts. Biometrika 61:383–385

Harville DA (1977) Maximum likelihood approaches to variance component estimation and to related problems (with discussion). J Am Stat Assoc 72:320–340

Harville DA (1997) Matrix algebra from a statistician's perspective. Springer, New York, NY

Hastie T, Tibshirani R, Friedman J (2001) The elements of statistical learning: data mining, inference, and prediction. Springer, New York, NY

Hayeck GT (2009) The kinematics of the upper extremity and subsequent effects on joint loading and surgical treatment. PhD thesis, Cornell University, Ithaca, NY

Hedayat A, Sloane N, Stufken J (1999) Orthogonal arrays. Springer, New York, NY

Helton JC (1993) Uncertainty and sensitivity analysis techniques for use in performance assessment for radioactive waste disposal. Reliab Eng Syst Saf 42:327–367

Helton JC, Johnson JD, Sallaberry CJ, Storlie CB (2006) Survey of sampling-based methods for uncertainty and sensitivity analysis. Reliab Eng Syst Saf 91:1175–1209

Hickernell FJ (1998) A generalized discrepancy and quadrature error bound. Math Comput 67:299–322

Higdon D, Kennedy M, Cavendish J, Cafeo J, Ryne R (2004) Combining field data and computer simulations for calibration and prediction. SIAM J Sci Comput 26:448–466

Higdon D, Gattiker J, Williams B, Rightley M (2008) Computer model calibration using high dimensional output. J Am Stat Assoc 103:570–583
Higdon DM (1998) A process-convolution approach to modelling temperatures in the North Atlantic Ocean (with discussion). Environ Ecol Stat 5:173–192
Higdon DM, Swall J, Kern JC (1999) Non-stationary spatial modeling. In: Bernardo JM, Berger JO, Dawid AP, Smith AFM (eds) Bayesian statistics, vol 6. Oxford University Press, Oxford, pp 761–768
Hoeffding W (1948) A class of statistics with asymptotically normal distribution. Ann Math Stat 19(3):293–325
Huber PJ (1981) Robust statistics. Wiley, New York, NY
Janssens ML (2000) Introduction to mathematical fire modeling, 2nd edn. Technomic Publishing Company, Lancaster
Jeffreys H (1961) Theory of probability. Oxford University Press, New York, NY
John JA (1987) Cyclic designs. Chapman & Hall, New York, NY
John PWM (1980) Incomplete block designs. M. Dekker, Inc., New York, NY
Johnson ME, Moore LM, Ylvisaker D (1990) Minimax and maximin distance designs. J Stat Plann Inf 26:131–148
Johnson RT, Jones B, Fowler JW, Montgomery DC (2008) Comparing designs for computer simulation experiments. In: Mason SJ, Hill RR, Moench L, Rose O, Jefferson T, Fowler JW (eds) Proceedings of the 2008 winter simulation conference, pp 463–470
Johnson R, Montgomery D, Jones B, Parker P (2010) Comparing computer experiments for fitting high order polynomial metamodels. J Qual Technol 42(1):86–102
Johnson RT, Montgomery DC, Jones B (2011) An empirical study of the prediction performance of space-filling designs. Int J Exp Design Process Optim 2(1):1–18
Jones B, Johnson R (2009) Design and analysis for the Gaussian process model. Qual Reliab Eng Int 25(5):515–524
Jones DR, Perttunen CD, Stuckman BE (1993) Lipschitzian optimization without the Lipschitz constant. J Optim Theory Appl 79(1):157–181
Jones DR, Schonlau M, Welch WJ (1998) Efficient global optimization of expensive black-box functions. J Glob Optim 13:455–492
Joseph VR, Hung Y, Sudjianto A (2008) Blind kriging: a new method for developing metamodels. ASME J Mech Des 130:377–381
Joseph VR, Gul E, Ba S (2015) Maximum projection designs for computer experiments. Biometrika 102(2):371–380
Journel AG, Huijbregts CJ (1978) Mining geostatistics. Academic, San Diego, CA
Kackar RN, Harville DA (1984) Approximations for standard errors of estimators of fixed and random effects in mixed linear models. J Am Stat Assoc 87:853–862
Kaufman C, Bingham D, Habib S, Heitmann K, Frieman J (2011) Efficient emulators of computer experiments using compactly supported correlation functions, with an application to cosmology. Ann Appl Stat 5:2470–2492
Keane AJ (2006) Statistical improvement criteria for use in multiobjective design optimization. AIAA J 44:879–891

Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of the 1995 IEEE international conference on neural networks, vol 4, pp 1942–1948

Kennedy MC, O'Hagan A (2000) Predicting the output from a complex computer code when fast approximations are available. Biometrika 87:1–13

Kennedy MC, O'Hagan A (2001) Bayesian calibration of computer models (with discussion). J R Stat Soc Ser B 63:425–464

Kleijnen JPC (1987) Review of random and group-screening designs. Commun Stat Theory Methods 16:2885–2900

Kleijnen JPC, Bettonvil B, Persson F (2006) Screening for the important factors in large discrete-event simulation models: sequential bifurcation and its applications. In: Dean AM, Lewis SM (eds) Screening: methods for experimentation in industry, drug discovery and genetics. Springer, New York, NY, pp 287–307

Kotz S, Nadarajah S (2004) Multivariate *t* distributions and their applications. Cambridge University Press, Cambridge

Kotzar GM, Davy DT, Berilla J, Goldberg VM (1995) Torsional loads in the early postoperative period following total hip replacement. J Orthop Res 13:945–955

Kreyszig E (1999) Advanced engineering mathematics. Wiley, New York, NY

Lagarias JC, Reeds JA, Wright MH, Wright PE (1998) Convergence properties of the Nelder-Mead simplex method in low dimensions. SIAM J Optim 8:112–147

Lam CQ, Notz WI (2008) Sequential adaptive designs in computer experiments for response surface model fit. Stat Appl 6:207–233

Leatherman ER (2013) Optimal predictive designs for experiments that involve computer simulators. PhD thesis, Department of Statistics, The Ohio State University, Columbus, OH

Leatherman ER, Dean AM, Santner TJ (2014) Computer experiment designs via particle swarm optimization. In: Melas VB, Mignani S, Monari P, Salmaso L (eds) Topics in statistical simulation: research papers from the 7th international workshop on statistical simulation, vol 114. Springer, Berlin, pp 309–317

Leatherman ER, Dean AM, Santner TJ (2017) Designing combined physical and computer experiments to maximize prediction accuracy. Comput Stat Data Anal 113:346–362

Leatherman ER, Santner TJ, Dean AM (2018) Computer experiment designs for accurate prediction. Stat Comput 28:739–751

Lehman J, Santner TJ, Notz WI (2004) Design of computer experiments to determine robust control variables. Stat Sinica 14:571–580

Lemieux C (2009) Monte Carlo and Quasi-Monte Carlo sampling. Springer, New York, NY

Lempert R, Williams BJ, Hendrickson J (2002) Using sensitivity analysis to support robust adaptive planning. Technical report, RAND

Lewis SM, Dean AM (2001) Detection of interactions in experiments on large numbers of factors (with discussion). J R Stat Soc Ser B 63:633–672

Li B, Genton MG, Sherman M (2008) Testing the covariance structure of multivariate random fields. Biometrika 95:813–829

Liefvendahl M, Stocki R (2006) A study on algorithms for optimization of Latin hypercubes. J Stat Plann Inf 136:3231–3247

Lindley DV (1956) On a measure of information provided by an experiment. Ann Math Stat 27:986–1005

Linkletter C, Bingham D, Hengartner N, Higdon D, Ye KQ (2006) Variable selection for Gaussian process models in computer experiments. Technometrics 48:478–490

Liu JS (2008) Monte Carlo strategies in scientific computing. Springer, New York, NY

Loeppky JL, Sacks J, Welch WJ (2009) Choosing the sample size of a computer experiment: a practical guide. Technometrics 51(4):366–376

Loeppky JL, Moore LM, Williams BJ (2010) Batch sequential designs for computer experiments. J Stat Plann Inf 140(6):1452–1464

Loeppky JL, Moore LM, Williams BJ (2012) Projection array based designs for computer experiments. J Stat Plann Inf 142:1493–1505

Loeppky JL, Williams BJ, Moore LM (2013) Global sensitivity analysis for mixture experiments. Technometrics 55:68–78

Loh WL (1996) On Latin hypercube sampling. Ann Stat 24:2058–2080

Lynn RR (1997) Transport model for prediction of wildfire behavior. Technical report LA-13334-T, Los Alamos National Laboratory

MacDonald B, Ranjan P, Chipman H (2015) GPfit: an R package for Gaussian process model fitting using a new optimization algorithm. J Stat Softw 64(12):1–23

Marrel A, Iooss B, Lauren B, Roustant O (2009) Calculations of Sobol indices for the Gaussian process metamodel. Reliab Eng Syst Saf 94:742–751

Matérn B (1960) Spatial variation. PhD thesis, Meddelanden fran Statens Skogsforskningsinstitut, vol 49, Num 5

Matérn B (1986) Spatial variation, 2nd edn. Springer, New York, NY

Matheron G (1963) Principles of geostatistics. Econ Geol 58:1246–1266

McKay MD, Beckman RJ, Conover WJ (1979) A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 21:239–245

McMillan NJ, Sacks J, Welch WJ, Gao F (1999) Analysis of protein activity data by Gaussian stochastic process models. J Biopharm Stat 9:145–160

Mease D, Bingham D (2006) Latin hyperrectangle sampling for computer experiments. Technometrics 48:467–477

Mitchell TJ (1974) An algorithm for the construction of "D-optimal" experimental designs. Technometrics 16:203–210

Mitchell TJ, Scott DS (1987) A computer program for the design of group testing experiments. Commun Stat Theory Methods 16:2943–2955

Mitchell TJ, Morris MD, Ylvisaker D (1990) Existence of smoothed stationary processes on an interval. Stoch Process Appl 35:109–119

Mitchell TJ, Morris MD, Ylvisaker D (1994) Asymptotically optimum experimental designs for prediction of deterministic functions given derivative information. J Stat Plann Inf 41:377–389

Mitchell MW, Genton MG, Gumpertz ML (2005) Testing for separability of space-time covariances. Environmetrics 16:819–831

Mockus A (1998) Estimating dependencies from spatial averages. J Comput Graph Stat 7:501–513

Mockus J, Tiešis V, Žilinskas A (1978) The application of Bayesian methods for seeking the extremum. In: Dixon LCW, Szego GP (eds) Towards global optimisation, vol 2. North Holland, Amsterdam, pp 117–129

Mockus A, Mockus J, Mockus L (1994) Bayesian approach adapting stochastic and heuristic methods of global and discrete optimization. Informatica 5:123–166

Mockus J, Eddy W, Mockus A, Mockus L, Reklaitis G (1997) Bayesian heuristic approach to discrete and global optimization: algorithms, visualization, software, and applications. Kluwer Academic, New York, NY

Montgomery GP, Truss LT (2001) Combining a statistical design of experiments with formability simulations to predict the formability of pockets in sheet metal parts. Technical report 2001-01-1130, Society of Automotive Engineers

Moon H (2010) Design and analysis of computer experiments for screening input variables. PhD thesis, Department of Statistics, The Ohio State University, Columbus, OH

Moon H, Santner TJ, Dean AM (2012) Two-stage sensitivity-based group screening in computer experiments. Technometrics 54(4):376–387

Morris MD (1991) Factorial sampling plans for preliminary computational experiments. Technometrics 33:161–174

Morris MD (2006) An overview of group factor screening. In: Dean AM, Lewis SM (eds) Screening: methods for experimentation in industry, drug discovery and genetics. Springer, New York, NY, pp 191–206

Morris MD (2012) Gaussian surrogates for computer models with time-varying inputs and outputs. Technometrics 54(1):42–50

Morris MD (2014) Maximin distance optimal designs for computer experiments with time-varying inputs and outputs. J Stat Plann Inf 144:63–68

Morris MD, Mitchell TJ (1995) Exploratory designs for computational experiments. J Stat Plann Inf 43:381–402

Morris MD, Mitchell TJ, Ylvisaker D (1993) Bayesian design and analysis of computer experiments: use of derivatives in surface prediction. Technometrics 35:243–255

Morris MD, Moore LM, McKay MD (2006) Sampling plans based on balanced incomplete block designs for evaluating the importance of computer model inputs. J Stat Plann Inf 136(9):3203–3220

Morris MD, Moore LM, McKay MD (2008) Using orthogonal arrays in the sensitivity analysis of computer models. Technometrics 50(2):205–215

Nelder JA, Mead R (1965) A simplex method for function minimization. Comput J 7:308–313

Niederreiter H (1988) Low-discrepancy and low-dispersion sequences. J Number Theory 30:51–70

Niederreiter H (1992) Random number generation and quasi-Monte Carlo methods. SIAM, Philadelphia, PA

Nocedal J, Wright S (2006) Numerical optimization. Springer series in operations research and financial engineering. Springer, New York, NY

Notz WI (2015) Expected improvement designs. In: Bingham D, Dean AM, Morris M, Stufken J (eds) Handbook of design and analysis of experiments. Chapman and Hall, New York, NY, pp 675–716

Oakley JE (2002) Eliciting Gaussian process priors for complex computer codes. Statistician 51:81–97

Oakley JE (2009) Decision-theoretic sensitivity analysis for complex computer models. Technometrics 5(2):121–129

Odeh R, Davenport J, Pearson N (eds) (1988) Selected tables in mathematical statistics, vol 11. American Mathematical Society, Providence, RI

O'Hagan A (1994) Kendall's advanced theory of statistics. Volume 2B: Bayesian inference. Cambridge University Press, Cambridge

O'Hagan A, Haylock RG (1997) Bayesian uncertainty analysis and radiological protection. In: Barnett V, Turkman KF (eds) Statistics for the environment, vol 3. Wiley, New York, NY, pp 109–128

O'Hagan A, Kennedy MC, Oakley JE (1999) Uncertainty analysis and other inference tools for complex computer codes. In: Bernardo JM, Berger JO, Dawid AP, Smith AFM (eds) Bayesian statistics, vol 6. Oxford University Press, Oxford, pp 503–524

Ong K, Santner T, Bartel D (2008) Robust design for acetabular cup stability accounting for patient and surgical variability. J Biomech Eng 130(031001):1–11

Overstall AM, Woods DC (2016) Multivariate emulation of computer simulators: model selection and diagnostics with application to a humanitarian relief model. J R Stat Soc Ser C 65(4):483–505

Owen AB (1992a) A central limit theorem for Latin hypercube sampling. J R Stat Soc Ser B Methodol 54:541–551

Owen AB (1992b) Orthogonal arrays for computer experiments, integration and visualization. Stat Sinica 2:439–452

Owen AB (1994) Controlling correlations in Latin hypercube samples. J Am Stat Assoc 89:1517–1522

Owen AB (1995) Randomly permuted (t, m, s)-nets and (t, s) sequences. In: Niederreiter H, Shiue PJS (eds) Monte Carlo and quasi-Monte Carlo methods in scientific computing. Springer, New York, NY, pp 299–317

Park JS (1994) Optimal Latin-hypercube designs for computer experiments. J Stat Plann Inf 39:95–111

Parzen E (1962) Stochastic processes. Holden-Day, San Francisco, CA

Patterson HD, Thompson R (1971) Recovery of interblock information when block sizes are unequal. Biometrika 58:545–554

Patterson HD, Thompson R (1974) Maximum likelihood estimation of components of variance. In: Proceedings of the 8th international biometric conference, Washington DC, pp 197–207

Paulo R, García-Donato G, Palomo J (2012) Calibration of computer models with multivariate output. Comput Stat Data Anal 56:3959–3974

Pebesma EJ, Heuvelink GBM (1999) Latin hypercube sampling of Gaussian random fields. Technometrics 41:303–312

Phoa FKH, Chen RB, Wang W, Wong WK (2015) Optimizing two-level supersaturated designs using swarm intelligence techniques. Technometrics 58(1):43–49

Picard R, Williams B (2013) Rare event estimation for computer models. Am Stat 67(1):22–32

Picheny V, Ginsbourger D, Richet Y, Caplin G (2013) Quantile-based optimization of noisy computer experiments with tunable precision. Technometrics 55(1):2–13

Plumlee M (2017) Bayesian calibration of inexact computer models. J Am Stat Assoc 112(519):1274–1285

Pratola M, Higdon D (2016) Bayesian additive regression tree calibration of complex high-dimensional computer models. Technometrics 58(2):166–179

Press WH, Teukolsky SA, Vetterling WT, Flannery BP (2007) Numerical recipes 3rd edition: the art of scientific computing. Cambridge University Press, New York, NY

Preston DL, Tonks DL, Wallace DC (2003) Model of plastic deformation for extreme loading conditions. J Appl Phys 93:211–220

Pujol G (2009) Simplex-based screening designs for estimating metamodels. Reliab Eng Syst Saf 94:1156–1160

Pukelsheim F (1993) Optimal design of experiments. Wiley, New York, NY

Qian PZ, Seepersad CC, Joseph VR, Allen JK, Wu CFJ (2006) Building surrogate models with details and approximate simulations. ASME J Mech Des 128:668–677

Qian PZG (2009) Nested Latin hypercube designs. Biometrika 96:957–970

Qian PZG (2012) Sliced Latin hypercube designs. J Am Stat Assoc 107:393–399

Qian PZG, Wu CFJ (2008) Bayesian hierarchical modeling for integrating low-accuracy and high-accuracy experiments. Technometrics 50(2):192–204

Qian PZG, Wu H, Wu CFJ (2008) Gaussian Process models for computer experiments with qualitative and quantitative factors. Technometrics 50(3):383–396

Raghavarao D (1971) Constructions and combinatorial problems in design of experiments. Wiley, New York, NY

Ranjan P, Bingham D, Michailidis G (2008) Sequential experiment design for contour estimation from complex computer codes. Technometrics 50(4):527–541

Reese CS, Wilson AG, Hamada M, Martz HF, Ryan KJ (2004) Integrated analysis of computer and physical experiments. Technometrics 46(2):153–164

Regis RG, Shoemaker CA (2005) Constrained global optimization of expensive black box functions using radial basis functions. J Glob Optim 31:153–171

Reich BJ, Storlie CB, Bondell HD (2009) Variable selection in Bayesian smoothing spline ANOVA models: application to deterministic computer codes. Technometrics 5(2):110–120

Rinnooy Kan AHG, Timmer GT (1984) A stochastic approach to global optimization. In: Boggs PT, Byrd RH, Schnabel RB (eds) Optimization 84: proceedings of the SIAM conference on numerical optimization. SIAM, Philadelphia, PA pp 245–262

Ripley BD (1981) Spatial statistics. Wiley, New York, NY

Roberts GO, Rosenthal JS (2007) Coupling and ergodicity of adaptive Markov chain Monte Carlo algorithms. J Appl Probab 44:458–475

Roberts GO, Smith AFM (1994) Simple conditions for the convergence of the Gibbs sampler and Metropolis-Hastings algorithms. Stoch Process Appl 49:207–216

Rodríguez-Iturbe I, Mejía JM (1974) The design of rainfall networks in time and space. Water Resour Res 10:713–728

Rougier J (2007) Lightweight emulators for multivariate deterministic functions. Department of Mathematics, University of Bristol

Roy S, Notz WI (2014) Estimating percentiles in computer experiments: a comparison of sequential-adaptive designs and fixed designs. Stat Theory Practice 8:12–29

Russi TM (2010) Uncertainty quantification with experimental data and complex system models. PhD thesis, UC Berkeley, CA

Sacks J, Schiller SB, Welch WJ (1989a) Design for computer experiments. Technometrics 31:41–47

Sacks J, Welch WJ, Mitchell TJ, Wynn HP (1989b) Design and analysis of computer experiments. Stat Sci 4:409–423

Sahama AR, Diamond NT (2001) Sample size considerations and augmentation of computer experiments. J Stat Comput Simul 68(4):307–319

Sahama TR, Diamond NT (2008) Computer experiment – a case study for modelling and simulation of manufacturing systems. Asian Int J Sci Technol Prod Manuf 1(2):97–103

Saltelli A (2002) Making best use of model evaluations to compute sensitivity indices. Comput Phys Commun 145 (2):280–297

Saltelli A, Sobol′ IM (1995) About the use of rank transformation in sensitivity analysis of model output. Reliab Eng Syst Saf 50:225–239

Saltelli A, Chan K, Scott E (2000) Sensitivity analysis. Wiley, Chichester

Saltelli A, Tarantola S, Campolongo F, Ratto M (2004) Sensitivity analysis in practice: a guide to assessing scientific models. Wiley, Chichester

Saltelli A, Annoni P, Azzini I, Campolongo F, Ratto M, Tarantola S (2010) Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index. Comput Phys Commun 181:259–270

Sampson PD, Guttorp P (1992) Nonparametric estimation of nonstationary covariance structure. J Am Stat Assoc 87:108–119

Savitsky T, Vannucci M, Sha N (2011) Variable selection for nonparametric Gaussian process priors: models and computational strategies. Stat Sci 26(1):130–149

Schonlau M (1997) Computer experiments and global optimization. PhD thesis, Department of Statistics and Actuarial Science, University of Waterloo, Waterloo, ON

Schonlau M, Welch WJ, Jones DR (1998) Global versus local search in constrained optimization of computer models. In: Flournoy N, Rosenberger WF, Wong WK (eds) New developments and applications in experimental design, vol 34. Institute of Mathematical Statistics, pp 11–25

Shannon CE (1948) A mathematical theory of communication. Bell Syst Tech J 27:379–423

Shewry MC, Wynn HP (1987) Maximum entropy sampling. J Appl Stat 14:165–170

Silvestrini RT, Montgomery DC, Jones B (2013) Comparing computer experiments for the Gaussian process model using integrated prediction variance. Qual Eng 25:164–174

Silvey SD (1980) Optimal design: an introduction to the theory for parameter estimation. Chapman & Hall, New York, NY

Sobol´ IM (1967) On the distribution of points in a cube and the approximate evaluation of integrals. USSR Comput Math Math Phys 7(4):86–112

Sobol´ IM (1976) Uniformly distributed sequences with an additional uniform property. USSR Comput Math Math Phys 16(5):236–242

Sobol´ IM (1990) Sensitivity estimates for nonlinear mathematical models. Matematicheskoe Modelirovanie 2:112–118

Sobol´ IM (1993) Sensitivity estimates for nonlinear mathematical models. Math Model Comput Exp 1(4):407–414

Sobol´ I, Kucherenko S (2009) Derivative based global sensitivity measures and their link with global sensitivity indices. Math Comput Simul 79(10):3009–3017

Stein ML (1987) Large sample properties of simulations using Latin hypercube sampling. Technometrics 29:143–151

Stein ML (1999) Interpolation of spatial data: some theory for kriging. Springer, New York, NY

Stein ML (2008) A modeling approach for large spatial datasets. J Korean Stat Soc 37:3–10

Stinstra E, den Hertog D, Stehouwer P, Vestjens A (2003) Constrained maximin designs for computer experiments. Technometrics 45(4):340–346

Stewart GW (1993) On the early history of the singular value decomposition. SIAM Rev 35(4):551–566

Stone M (1974) Cross-validatory choice and assessment of statistical predictions (with discussion) (correction: 38:102). J R Stat Soc Ser B 36:111–147

Stone M (1977) An asymptotic equivalence of choice of model by cross-validation and Akaike's criterion. J R Stat Soc Ser B 39:44–47

Storlie CB, Helton JC (2008) Multiple predictor smoothing methods for sensitivity analysis: description of technique. Reliab Eng Syst Saf 93:28–54

Storlie CB, Swiler LP, Helton JC, Salaberry CJ (2009) Implementation and evaluation of nonparametric regression procedures for sensitivity analysis of computationally demanding models. Reliab Eng Syst Saf 94(11):1735–1763

Storlie CB, Lane WA, Ryan EM, Gattiker JR, Higdon DM (2015) Calibration of computational models with categorical parameters and correlated outputs via Bayesian smoothing spline ANOVA. J Am Stat Assoc 110(509):68–82

Street AP, Street DJ (1987) Combinatorics of experimental design. Oxford University Press, Oxford

Stripling HF, McCarren RG, Kuranz CC, Grosskopf MJ, Rutter E, Torralva BR (2013) A calibration and data assimilation method using the Bayesian MARS emulator. Ann Nucl Energy 52:103–112

Sun F, Santner TJ, Dean AM (2013) One-at-a-time designs for estimating elementary effects of simulator experiments with non-rectangular input regions. Stat Appl 11:15–32

Svenson JD (2011) Computer experiments: multiobjective optimization and sensitivity analysis. PhD thesis, Department of Statistics, The Ohio State University, Columbus, OH

Svenson J, Santner T (2016) Multiobjective optimization of expensive-to-evaluate deterministic computer simulator models. Comput Stat Data Anal 94:250–264

Svenson J, Santner TJ, Dean A, Moon H (2014) Estimating sensitivity indices based on Gaussian process metamodels with compactly supported correlation functions. J Stat Plann Inf 144:160–172

Tan MHY (2013) Minimax designs for finite design regions. Technometrics 55:346–358

Tang B (1993) Orthogonal array-based Latin hypercubes. J Am Stat Assoc 88:1392–1397

Tang B (1994) A theorem for selecting OA-based Latin hypercubes using a distance criterion. Commun Stat Theory Methods 23:2047–2058

Trosset MW (1999) Approximate maximin distance designs. In: ASA Proceedings of the section on physical and engineering sciences. American Statistical Association, Alexandria, VA, pp 223–227

Trosset MW, Padula AD (2000) Designing and analyzing computational experiments for global optimization. Technical report 00-25, Department of Computational & Applied Mathematics, Rice University

Trunin RF (1998) Shock compression of condensed materials. Cambridge University Press, Cambridge

Tuo R, Wu CFJ (2016) A theoretical framework for calibration in computer models: parametrization, estimation and convergence properties. SIAM/ASA Int J Uncertain Quantif 4(1):767–795

Van Der Vaart AW (1998) Asymptotic statistics. Cambridge University Press, Cambridge

Vazquez E, Bect J (2010) Pointwise consistency of the kriging predictor with known mean and covariance functions. In: Giovagnoli A, Atkinson AC, Torsney B, May C (eds) mODa 9 - advances in model-oriented design and analysis. Physica-Verlag, Heidelberg, pp 221–228

Vazquez E, Bect J (2011) Sequential search based on kriging: convergence analysis of some algorithms. Proceedings of the 58th world statistical congress of the ISI, pp 1241–1250

Vecchia AV (1988) Estimation and identification for continuous spatial processes. J R Stat Soc Ser B 50:297–312

Ver Hoef JM, Barry RP (1998) Constructing and fitting models for cokriging and multivariable spatial prediction. J Stat Plann Inf 69:275–294

Vicario G, Craparotta G, Pistone G (2016) Metamodels in computer experiments: kriging versus artificial neural networks. Qual Reliab Eng Int 32:2055–2065

Villarreal-Marroquín MG, Chen PH, Mulyana R, Santner TJ, Dean AM, Castro JM (2017) Multiobjective optimization of injection molding using a calibrated predictor based on physical and simulated data. Polym Eng Sci 57(3):248–257

Vine AE, Lewis SM, Dean AM (2005) Two-stage group screening in the presence of noise factors and unequal probabilities of active effects. Stat Sinica 15:871–888

Wahba G (1980) Spline bases, regularization, and generalized cross validation for solving approximation problems with large quantities of noisy data. In: Cheney EW (ed) Approximation theory III. Academic, New York, NY, pp 905–912

Wallstrom TC (2007) Estimating replicate variation. Technical report LA-UR-07-4412, Los Alamos National Laboratory, Los Alamos, NM

Walton W (1985) ASET-B: a room fire program for personal computers. Technical Report 85-3144-1, National Bureau of Standards, Washington

Weaver BP, Williams BJ, Anderson-Cook CM, Higdon DM (2016) Computational enhancements to Bayesian design of experiments using Gaussian processes. Bayesian Anal 11(1):191–213

Welch WJ (1985) ACED: algorithms for the construction of experimental designs. Am Stat 39:146

Welch WJ, Buck RJ, Sacks J, Wynn HP, Mitchell TJ, Morris MD (1992) Screening, predicting, and computer experiments. Technometrics 34:15–25

Wiens DP (1991) Designs for approximately linear regression: Two optimality properties of uniform designs. Stat Probab Lett 12:217–221

Williams BJ, Santner TJ, Notz WI (2000) Sequential design of computer experiments to minimize integrated response functions. Stat Sinica 10:1133–1152

Williams BJ, Santner TJ, Notz WI, Lehman JS (2010) Sequential design of computer experiments for constrained optimization. In: Kneib T, Tutz G (eds) Statistical modelling and regression structures: Festschrift in Honour of Ludwig Fahrmeir. Physica-Verlag, Heidelberg, pp 449–472

Williams BJ, Loeppky JL, Moore LM, Macklem MS (2011) Batch sequential design to achieve predictive maturity with calibrated computer models. Reliab Eng Syst Saf 96(9):1208–1219

Worley BA (1987) Deterministic uncertainty analysis. Technical report ORNL-6428, National Technical Information Service, 5285 Port Royal Road, Springfield, VA 22161

Wu CFJ, Hamada M (2009) Experiments: planning, analysis, and parameter design optimization, 2nd edn. Wiley, New York, NY

Xiong Y, Chen W, Apley D, Ding X (2007) A non-stationary covariance-based kriging method for metamodelling in engineering design. Int J Numer Methods Eng 71(6):733–756

Yaglom AM (1962) Introduction to the theory of stationary random functions. Dover, New York, NY

Yaglom AM (1986) Correlation theory of stationary and related random functions (Volume II). Springer, New York, NY

Yang XS (2010) Engineering optimization: an introduction with metaheuristic applications. Wiley, Hoboken, NJ

Ye KQ (1998) Orthogonal column Latin hypercubes and their application in computer experiments. J Am Stat Assoc 93:1430–1439

Ye KQ, Li W, Sudjianto A (2000) Algorithmic construction of optimal symmetric Latin hypercube designs. J Stat Plann Inf 90(1):145–159

Zhang Y (2014) Computer experiments with both quantitative and qualitative inputs. PhD thesis, Department of Statistics, The Ohio State University, Columbus, OH

Zhou H (ed) (2013) Computer modeling for injection molding: simulation, optimization, and control. Wiley, Hoboken, NJ

Zhou Q, Qian PZG, Wu H, Zhou S (2011) A simple approach to emulation for computer models with qualitative and quantitative factors. Technometrics 53:266–273

Zimmerman DL, Cressie NA (1992) Mean squared prediction error in the spatial linear model with estimated covariance parameters. Ann Inst Stat Math 44:27–43

Zitzler E, Knowles J, Thiele L (2008) Quality assessment of Pareto set approximations. In: Branke J, Deb K, Miettinen K, Slowinski R (eds) Multiobjective optimization: interactive and evolutionary approaches. Springer, Berlin, pp 373–404

Author Index

Subject Index