Computer Experiments: Multiobjective Optimization and Sensitivity Analysis

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Joshua D. Svenson, B.S., M.S.

Graduate Program in Statistics

The Ohio State University

2011

Dissertation Committee:

Thomas J. Santner, Adviser
Angela M. Dean
William I. Notz
© Copyright by
Joshua D. Svenson
2011
ABSTRACT

Computer experiments have emerged as a popular tool for studying the relationship between a response variable and factors that affect the response when a computational model of this relationship is available. They have proven to be particularly useful in applications where properly designed physical experiments are infeasible. This thesis considers two problems that occur in the design and analysis of computer experiments. The first problem is approximation of the Pareto front and set in a multiple-output computer experiment. The second problem extends the calculation of sensitivity indices of input factors to a broader class of models than have been studied previously.

To solve the first problem, several new design criteria for approximating the Pareto front are developed. The resulting sequential designs generalize the well-known expected improvement approach for optimization of a single-objective function. The new methods are compared to previously proposed expected improvement generalizations for multiobjective optimization from the literature. The comparisons are based on both theoretical considerations and empirical results from using the sequential design criteria on several test functions and engineering applications.

In the sensitivity index problem, formulas are derived for calculating empirical Bayesian estimates of sensitivity indices for Gaussian process models with an arbitrary polynomial mean structure and three parametric correlation families. The use of
a polynomial mean has the potential to provide more accurate estimates of sensitivity indices when the computer output has a large-scale polynomial trend. Additionally, when combined with a compactly supported correlation function and parameter space restrictions that force a particular degree of sparsity on the correlation matrix at the design, the polynomial mean assumption allows one to estimate sensitivity indices for computer experiments with a large number of runs. In such large-design applications, estimates based on the standard constant mean Gaussian processes with a power exponential correlation function can be computationally infeasible. Examples are presented that exhibit the accuracy of the estimates under these nonstandard modeling assumptions.
To Amy, Mom, and Dad
ACKNOWLEDGMENTS

First and foremost, I would like to thank my wife Amy for staying by my side during a rewarding yet stressful five years of graduate school. Being married to a Ph.D. student is not the easiest job in the world, nor the most lucrative, but she has always offered her love and support, especially during the difficult times. Words will never be able to express how important you are to me. You are an awesome wife, but you are an even better friend. Thanks for making me feel like the luckiest guy in the world!

I also would like to thank my parents, John and Cathy Svenson, for being my biggest supporters, from the moment I was born until to the moment I finally receive my degree. You have always made it possible for me to pursue my dreams, with both moral and financial support, and for this, I am truly appreciative. I know you both have sacrificed a lot so that I could get where I am today. Although I'll never be able to repay you for everything you’ve done for me, I hope that the man I’ve become makes you feel that your sacrifices on my behalf have been worthwhile.

I am also lucky to have great friends and family that have shown me great support for the last five years. In particular, thanks to my brother Brian, for keeping my sane during my first few years of graduate school, and thanks to my brother Andy, who has always been there to help his younger brothers. Also, thanks to my cousins Brad, Courtney, and Amanda, who have helped make my life in Columbus a more enjoyable
one. Thanks to my whole extended family, both Svensons and Cramers, for always showing love and support. A owe a big thank you to my wife’s family in Willoughby. Arnie, Susie, Jen, Tommy, Jack and everyone else has always treated me like their own, and I appreciate that. You all have become not only my family, but also great friends. A guy couldn’t ask for better in-laws! A big shout-out to John Phillips, whose friendship has served as a welcome distraction to the rigors of graduate school. Also, thanks to my other best friends from high school, Chris Hawkins and Justin Shumney, for being great friends and supporters for as long as I’ve know them. Finally, thanks to all the support from my friends at Baldwin-Wallace, particularly Mark and Coree Miracola, Eric Schmuhl, Alex Johnson, Adam Eltrich, John Bacon, Ian Formaz, and Jon Shoffstal.

I owe a huge debt of gratitude to my advisor, Tom Santner, for his advice and support during the last three years. I am thankful for his special topics class in computer experiments, which ignited my interest in the field three years ago. His insight and guidance have kept me focused, on-track, and made the process of writing my dissertation as painless as possible. I am very appreciative of his open-door policy, which has allowed me to come to him with research questions at all hours of the day. I also appreciate the R.A. opportunities Dr. Santner has given to me, which have exposed me to several application areas, and made me a more well-rounded statistician. A Ph.D. student could not ask for a better advisor!

I would also like to thank Angela Dean for serving on my thesis and candidacy exam committees, for encouraging me to take part at my first statistics conference in Missouri, and for taking time out of her schedule to supervise my reading hours in advanced experimental design. It has truly been a pleasure to work with you
and Dr. Santner! I am greatly indebted to Bill Notz, who has also served on my thesis and candidacy exam committees, and who has also offered great insight and support while I was working as an R.A. on the EESEE project. Additionally, his daily derisions with regards to all things Cleveland Browns and Columbus Blue Jackets has served as a welcome distraction to the stressful process of earning a Ph.D. I would also like to thank Chris Hans for serving on my candidacy exam committee and for taking part in some insightful reading hours in Markov Chain Monte Carlo during my first year of graduate school. I would also like to extend a thank-you to Maria Villarreal-Marroquin and Jose Castro in the ISE Department for all of their help with the injection molding example in Chapter 3.

I am thankful for the entire OSU Statistics department, which has proven to be a great place to study statistics. I am particularly thankful for the guidance, instruction, and support of Elizabeth Stasny, Douglas Wolfe, Jackie Miller, Mike Fligner, Steve MacEachern, Dennis Pearl, Tao Shi, Doug Critchlow, H.N. Nagaraja, Radu Herbei, Kate Calder, Yoon Lee and Mario Peruggia.

I would never have gotten where I am without the instruction and encouragement of the great mathematics and computer science instructors at Baldwin-Wallace College and Wadsworth High School. I owe a large portion of my success to Chungsim Han, David Calvis, Tim Riggle, Susan Penko, Ken Atchinson, Jodi Tims, Randy Molmen, Dick Little, and Berhane Ghaim. A huge thank you to Mrs. Barlow and Mrs. North at WHS for cultivating my first real interest in mathematics.

I have been lucky to make some great friends during my time at Ohio State. While I’ve had a great relationship with almost everyone I’ve been in contact with in the Statistics department, there have been several who have stood out. I’d like to
thank Denise Kalos, whose friendship, affinity for Aqua Teen Hunger Force, and trips to Chipotle helped me survive the stress of my second qualifying exam. I am also grateful to have met Nader Gemayel, a genuinely great guy who not only helped me land my first job, but also kept everyone in the department laughing with his brutal honesty and wisecracks. Additionally, I had the opportunity to become friends with Danel Draguljić, who sense of humor helped make a trip to the University of Missouri an enjoyable one. I would also like to thank Matthias Katzfuss, Michael Sonksen, Joanne Sklowdowski, Mary Hickey, Fangfang Sun, Jenny Williams, Tayler Blake, Adam Dallas, Erinn Kingman, Bill Darnieder, Nathan Cruze, Hang Kim, Stephen Bamattre, Jenný Brynjarsdóttir, Candace Berrett, Erin Leatherman, and Gang Han for their friendship during my time at OSU.
VITA

February 9, 1984 .......................... Born - Barberton, OH

2006 ........................................ B.S. Mathematics, Baldwin-Wallace College

2008 ........................................ M.S. Statistics, The Ohio State University

2007-present ............................... Graduate Teaching and Research Associate, The Ohio State University

PUBLICATIONS

Research Publications


FIELDS OF STUDY

Major Field: Statistics
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>Computer Experiments</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>Motivating Examples</td>
<td>3</td>
</tr>
<tr>
<td>1.2.1</td>
<td>Design of a Piezoelectric Bimorph Actuator</td>
<td>3</td>
</tr>
<tr>
<td>1.2.2</td>
<td>Oil-Field Simulator</td>
<td>4</td>
</tr>
<tr>
<td>1.3</td>
<td>Modeling Multiple-Output Computer Experiments</td>
<td>5</td>
</tr>
<tr>
<td>1.3.1</td>
<td>Predictive Distribution</td>
<td>6</td>
</tr>
<tr>
<td>1.3.2</td>
<td>Covariance Structure</td>
<td>12</td>
</tr>
<tr>
<td>1.4</td>
<td>Optimization of Single-Output Computer Experiments</td>
<td>22</td>
</tr>
<tr>
<td>1.5</td>
<td>Multiobjective Optimization</td>
<td>26</td>
</tr>
<tr>
<td>1.5.1</td>
<td>Basic Concepts and Terminology</td>
<td>26</td>
</tr>
<tr>
<td>1.5.2</td>
<td>Pareto Set Approximation Quality Indicators</td>
<td>27</td>
</tr>
<tr>
<td>1.5.3</td>
<td>Classical Approaches to Multiobjective Optimization</td>
<td>33</td>
</tr>
<tr>
<td>1.5.4</td>
<td>Multiobjective Evolutionary Algorithms (MOEAs)</td>
<td>34</td>
</tr>
<tr>
<td>1.5.5</td>
<td>The Expected Improvement Approach</td>
<td>37</td>
</tr>
</tbody>
</table>
4. Estimation of Sensitivity Indices Based on a Gaussian Process Model with a Polynomial Mean ........................................... 225

4.1 Gaussian Processes with a Polynomial Mean .................. 226
  4.1.1 Statistical Model ........................................ 226
  4.1.2 Prediction .............................................. 227
  4.1.3 Motivating the Polynomial Mean Assumption .......... 229

4.2 Estimation of Sensitivity Indices in the Gaussian Process Model . 239

4.3 Computing the Integrals for Three Correlation Functions .......... 252
  4.3.1 Gaussian Correlation Function ......................... 253
  4.3.2 Bohman Correlation Function .......................... 255
  4.3.3 Cubic Correlation Function .............................. 264

4.4 Examples .................................................. 269
  4.4.1 A Simple Two Dimensional Function .................. 269
  4.4.2 Borehole Function ...................................... 272

5. Conclusions, Discussion, Recommendations and Future Research ...... 277

5.1 Multiobjective Optimization of Computer Experiments ........... 277

5.2 Sensitivity Analysis of Computer Experiments .................... 287

Appendices:

A. Multivariate Modeling Derivations and Proofs .................. 290
  A.1 Proof of Theorem 1 ................................ 290
  A.2 Proof of Theorem 2 ................................ 294
  A.3 Proof that \( \hat{y}(x_0) \) and \( S(x_0) \) are equivalent to the BLUP and MSPE matrix when \( n_0 = 1 \) .............. 296

B. Proofs of Theorems Related to the Expected Maximin Improvement Function ....................................................... 299
  B.1 Proof of Theorem 4 ................................ 299
  B.2 Proof of Theorem 5 ................................ 302

C. Software Manuals ........................................ 308
  C.1 \texttt{NSDep} User’s Guide ............................... 308
    C.1.1 Functions Called by \texttt{NSDep} ................. 312
    C.1.2 Some Details Regarding REML Estimation ............ 313
    C.1.3 Example .................................. 317
C.2 MultiOpt User’s Guide ........................................... 321
  C.2.1 Functions Called by MultiOpt .............................. 329
  C.2.2 Some notes on the optimization of $QI(x)$ ............... 332
  C.2.3 Example .................................................. 333
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 ERMSPE’s for the WSNL function when we assume $A$ is lower triangular in the NLMC. Notice that predictive ability of the NLMC changes if we swap the order of $y_1(\cdot)$ and $y_2(\cdot)$.</td>
<td>17</td>
</tr>
<tr>
<td>1.2 ERMSPE’s for the WSNL function when we assume $A = LL^\top$ in the NLMC, where $L$ is lower triangular. Notice predictions are (practically) unchanged if we swap the order of $y_1(\cdot)$ and $y_2(\cdot)$ in the NLMC.</td>
<td>19</td>
</tr>
<tr>
<td>2.1 Summary of Desirable Properties for existing and proposed improvement functions</td>
<td>107</td>
</tr>
<tr>
<td>3.1 List of the expected improvement algorithms used to approximate the Pareto front. Notice that “Method” (QIM1 or QIM2) is not specified for the probability of improvement, as QIM1 and QIM2 are equivalent. Also, “Dependent” refers to a nonseparable LMC model (see Section 1.3.2).</td>
<td>112</td>
</tr>
<tr>
<td>3.2 Summary of quality indicators for five runs of each algorithm for the MOP2 problem. The initial design is a 10-point maximin LHD, and 10 points are added via the expected improvement algorithm.</td>
<td>116</td>
</tr>
<tr>
<td>3.3 Summary of quality indicators for five runs of each algorithm for the MOP2 problem. The initial design is a 20-point maximin LHD, and 10 points are added via the expected improvement algorithm.</td>
<td>117</td>
</tr>
<tr>
<td>3.4 Summary of quality indicators for five runs of each algorithm for the WSNL problem. The initial design is a 20-point maximin LHD, and 10 points are added via the expected improvement algorithm.</td>
<td>142</td>
</tr>
</tbody>
</table>
3.5 Summary of quality indicators for five runs of each algorithm for the WSNL problem. The initial design is a 20-point maximin LHD, and 10 points are added via the expected improvement algorithm.

3.6 Elements of $X$ and $Y^{40.2}$ used to construct the DGP function.

3.7 Summary of quality indicators for five runs of each algorithm for the DGP problem. The initial design is a 10-point maximin LHD, and 10 points are added via the expected improvement algorithm.

3.8 Summary of quality indicators in five runs of each algorithm for the DGP problem. The initial design is a 20-point maximin LHD, and 10 points are added via the expected improvement algorithm.

3.9 Summary of quality indicators in five runs of each algorithm for the DTLZ2 problem. The initial design is a 20-point maximin LHD, and 20 points are added via the expected improvement algorithm.

3.10 Summary of quality indicators in five runs of each algorithm for the DTLZ2 problem. The initial design is a 40-point maximin LHD, and 20 points are added via the expected improvement algorithm.

3.11 Summary of quality indicators for the ZLT1 problem. Algorithms were run only one time for each improvement criteria, using only independence models for the output.

3.12 Summary of quality indicators for the twelve expected improvement algorithms for the Nowacki beam problem. Algorithms were run only once for each improvement criteria, and independent output models were used. The hypervolume indicator uses the constrained marginal maximums of the two outputs (0.0125, 1.13 x 10^8) as the reference point $R$.

4.1 Estimated main and total effect sensitivity indices for (4.4.1) using all three correlation functions and both constant and quadratic mean structures, as well as the ERMSPEs for the predictor of (4.4.1) produced by each GP model.

4.2 Estimated sensitivity indices for the borehole function based on a 160-point space-filling Latin hypercube design, a constant mean GP model, and an unrestricted correlation parameter space.
4.3 Details of the various implementations for estimating sensitivity indices with the 400-point design. \( \alpha \) is the maximum possible proportion of off-diagonal elements of \( \Sigma_{sim}^Z \) that can be non-zero when restricting the parameters space of the correlation parameters as described in (4.1.7)-(4.1.9) (applicable for compactly supported correlation function only), ERMSPE is calculated using a set of 200 validations points, and Time measures how long it took to fit the model and obtain estimates of sensitivity indices on an Intel Core 2 Duo CPU 3.00 with 4 GB RAM. The estimates of the sensitivity indices can be found in Table 4.4.

4.4 Estimated sensitivity indices for the Borehole function for an initial 400-point space-filling Latin hypercube design. The details of the various correlation function and software choices can be found in Table 4.3.
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Squared prediction errors $(\hat{y}_1(x_i) - y_1(x_i))^2$, $1 \leq i \leq 2500$ in the NLMC minus the squared prediction errors $(\hat{y}_1(x_i) - y_1(x_i))^2$, $1 \leq i \leq 2500$ in the independence model for $y_1(\cdot)$ in the WSNL function.</td>
<td>18</td>
</tr>
<tr>
<td>1.2</td>
<td>Squared prediction errors $(\hat{y}_2(x_i) - y_2(x_i))^2$, $1 \leq i \leq 2500$ in the NLMC minus the squared prediction errors $(\hat{y}_2(x_i) - y_2(x_i))^2$, $1 \leq i \leq 2500$ in the independence model for $y_2(\cdot)$ in the WSNL function.</td>
<td>18</td>
</tr>
<tr>
<td>1.3</td>
<td>The circles form a 5-point set $A$ in the objective space, the square is the reference point $R$, and the area of the gray shaded region is $I_H(A, R)$.</td>
<td>29</td>
</tr>
<tr>
<td>2.1</td>
<td>Surface plot of $I_K(y(x))$ when $m = 2$. The black squares represent the current Pareto front.</td>
<td>60</td>
</tr>
<tr>
<td>2.2</td>
<td>Surface plot of $I_H(y)$ when $m = 2$ and $R = (0.9, 0.9)$. The black squares represent the current Pareto front.</td>
<td>63</td>
</tr>
<tr>
<td>2.3</td>
<td>Surface plot of $I_P(y(x))$ when $m = 2$. The black squares represent the current Pareto front.</td>
<td>69</td>
</tr>
<tr>
<td>2.4</td>
<td>Surface plot of $I_M(y(x))$ when $m = 2$. The black squares represent the current Pareto front.</td>
<td>71</td>
</tr>
<tr>
<td>2.5</td>
<td>Regions of integration for a three point Pareto front.</td>
<td>74</td>
</tr>
<tr>
<td>2.6</td>
<td>Surface plot of $I_{WH}(y)$ with $\zeta = 1$ when $m = 2$. The black squares represent the current Pareto front.</td>
<td>80</td>
</tr>
<tr>
<td>2.7</td>
<td>$A \preceq B$ but $I_K(A) &gt; I_K(B)$.</td>
<td>96</td>
</tr>
</tbody>
</table>
3.1 \( y_1(\cdot) \) in the MOP2 function ............................................. 114

3.2 \( y_2(\cdot) \) in the MOP2 function ............................................. 114

3.3 Dark black circles are the approximated true Pareto front for the MOP2 function ............................................. 115

3.4 Dark black circles are the approximated true Pareto set for the MOP2 function ............................................. 115

3.5 The final run of the five sequential designs for the MOP2 problem with an initial design of size 10 and 10 points added sequentially via EMMI-Ind (top), EMAX1-Ind (middle) and EMMI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. ............................................. 120

3.6 The final run of the five sequential designs for the MOP2 problem with an initial design of size 10 and 10 points added sequentially via EHI1-Ind (top), EWHI1-Ind (middle) and KEI-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. ............................................. 121

3.7 The final run of the five sequential designs for the MOP2 problem with an initial design of size 10 and 10 points added sequentially via EHI2-Ind (top), EWHI2-Ind (middle) and KEI1-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. ............................................. 122

3.8 The final run of the five sequential designs for the MOP2 problem with an initial design of size 10 and 10 points added sequentially via PI-Ind (top), ECI1-Ind (middle) and ECI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. ............................................. 123
3.9 The final run of the five sequential designs for the MOP2 problem with an initial design of size 10 and 10 points added sequentially via EECI1-Ind (top) and EECI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.  

3.10 The final run of the five sequential designs for the MOP2 problem with an initial design of size 10 and 10 points added sequentially via EMMI-Dep (top), EMAX1-Dep (middle) and EMMI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.  

3.11 The final run of the five sequential designs for the MOP2 problem with an initial design of size 10 and 10 points added sequentially via EHI1-Dep (top), EWHI1-Dep (middle) and KEI-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.  

3.12 The final run of the five sequential designs for the MOP2 problem with an initial design of size 10 and 10 points added sequentially via EHI2-Dep (top), EWHI2-Dep (middle) and KEI1-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.  

3.13 The final run of the five sequential designs for the MOP2 problem with an initial design of size 10 and 10 points added sequentially via PI-Dep (top) and ECI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.  

3.14 The final run of the five sequential designs for the MOP2 problem with an initial design of size 10 and 10 points added sequentially via EECI1-Dep (top) and EECI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
3.15 The final run of the five sequential designs for the MOP2 problem with an initial design of size 20 and 10 points added sequentially via EMMI-Ind (top), EMAXI-Ind (middle) and EMMI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. ................................................................. 130

3.16 The final run of the five sequential designs for the MOP2 problem with an initial design of size 20 and 10 points added sequentially via EHI1-Ind (top), EWHI1-Ind (middle) and KEI-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. ................................................................. 131

3.17 The final run of the five sequential designs for the MOP2 problem with an initial design of size 20 and 10 points added sequentially via EHI2-Ind (top), EWHI2-Ind (middle) and KEI1-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. ................................................................. 132

3.18 The final run of the five sequential designs for the MOP2 problem with an initial design of size 20 and 10 points added sequentially via PI-Ind (top), ECI1-Ind (middle) and ECI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. ................................................................. 133

3.19 The final run of the five sequential designs for the MOP2 problem with an initial design of size 20 and 10 points added sequentially via EECI1-Ind (top) and EECI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. . . . 134

3.20 The final run of the five sequential designs for the MOP2 problem with an initial design of size 20 and 10 points added sequentially via EMMI-Dep (top), EMAX1-Dep (middle) and EMMI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. ................................................................. 135
3.21 The final run of the five sequential designs for the MOP2 problem with an initial design of size 20 and 10 points added sequentially via EHI1-Dep (top), EWHI1-Dep (middle) and KEI-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.  ................................................. 136

3.22 The final run of the five sequential designs for the MOP2 problem with an initial design of size 20 and 10 points added sequentially via EHI2-Dep (top), EWHI2-Dep (middle) and KEI1-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.  ................................................. 137

3.23 The final run of the five sequential designs for the MOP2 problem with an initial design of size 20 and 10 points added sequentially via PI-Dep (top) and ECI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.  .................. 138

3.24 The final run of the five sequential designs for the MOP2 problem with an initial design of size 20 and 10 points added sequentially via EECI1-Dep (top) and EECI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.  .... 139

3.25 $y_1(\cdot)$ in the WSNL function ................................................. 140

3.26 $y_2(\cdot)$ in the WSNL function ................................................. 140

3.27 The dark black region is the approximated true Pareto front for the WSNL function  ................................................. 141

3.28 The dark black region is the the approximated true Pareto set for the WSNL function  ................................................. 141
3.29 The final run of the five sequential designs for the WSNL problem with an initial design of size 10 and 10 points added sequentially via EMMI-Ind (top), EMAX1-Ind (middle) and EMMI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. ................................................................. 147

3.30 The final run of the five sequential designs for the WSNL problem with an initial design of size 10 and 10 points added sequentially via EHI1-Ind (top), EWHI1-Ind (middle) and KEI-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. ................................................................. 148

3.31 The final run of the five sequential designs for the WSNL problem with an initial design of size 10 and 10 points added sequentially via EHI2-Ind (top), EWHI2-Ind (middle) and KEI1-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. ................................................................. 149

3.32 The final run of the five sequential designs for the WSNL problem with an initial design of size 10 and 10 points added sequentially via PI-Ind (top), ECI1-Ind (middle) and ECI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. ................................................................. 150

3.33 The final run of the five sequential designs for the WSNL problem with an initial design of size 10 and 10 points added sequentially via EECI1-Ind (top) and EECI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. ................................................................. 151

3.34 The final run of the five sequential designs for the WSNL problem with an initial design of size 10 and 10 points added sequentially via EMMI-Dep (top), EMAX1-Dep (middle) and EMMI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. ................................................................. 152
The final run of the five sequential designs for the WSNL problem with an initial design of size 10 and 10 points added sequentially via EHI1-Dep (top), EWH11-Dep (middle) and KEI-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.

The final run of the five sequential designs for the WSNL problem with an initial design of size 10 and 10 points added sequentially via EHI2-Dep (top), EWH12-Dep (middle) and KEI1-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.

The final run of the five sequential designs for the WSNL problem with an initial design of size 10 and 10 points added sequentially via PI-Dep (top) and ECI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.

The final run of the five sequential designs for the WSNL problem with an initial design of size 10 and 10 points added sequentially via EECI1-Dep (top) and EECI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.

The final run of the five sequential designs for the WSNL problem with an initial design of size 20 and 10 points added sequentially via EMMI-Ind (top), EMAX1-Ind (middle) and EMMI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.

The final run of the five sequential designs for the WSNL problem with an initial design of size 20 and 10 points added sequentially via EHI1-Ind (top), EWH11-Ind (middle) and KEI-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
3.41 The final run of the five sequential designs for the WSNL problem with an initial design of size 20 and 10 points added sequentially via EHI2-Ind (top), EWHI2-Ind (middle) and KEI1-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. ......................................................... 159

3.42 The final run of the five sequential designs for the WSNL problem with an initial design of size 20 and 10 points added sequentially via PI-Ind (top), ECI1-Ind (middle) and ECI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. ......................................................... 160

3.43 The final run of the five sequential designs for the WSNL problem with an initial design of size 20 and 10 points added sequentially via EECI1-Ind (top) and EECI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. ......................................................... 161

3.44 The final run of the five sequential designs for the WSNL problem with an initial design of size 20 and 10 points added sequentially via EMMI-Dep (top), EMAX1-Dep (middle) and EMMI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. ......................................................... 162

3.45 The final run of the five sequential designs for the WSNL problem with an initial design of size 20 and 10 points added sequentially via EHI1-Dep (top), EWHI1-Dep (middle) and KEI-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. ......................................................... 163

3.46 The final run of the five sequential designs for the WSNL problem with an initial design of size 20 and 10 points added sequentially via EHI2-Dep (top), EWHI2-Dep (middle) and KEI1-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. ......................................................... 164
3.47 The final run of the five sequential designs for the WSNL problem with an initial design of size 20 and 10 points added sequentially via PI-Dep (top) and ECI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.  

3.48 The final run of the five sequential designs for the WSNL problem with an initial design of size 20 and 10 points added sequentially via EECI1-Dep (top) and EECI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.

3.49 $y_1(\cdot)$ in the $DGP_2$ function.

3.50 $y_2(\cdot)$ in the $DGP$ function.

3.51 Dark black circles are the approximated true Pareto front for the $DGP$ function.

3.52 Dark black circles are the approximated true Pareto set for the $DGP$ function.

3.53 The final run of the five sequential designs for the DGP problem with an initial design of size 10 and 10 points added sequentially via EMMI-Ind (top), EMAX1-Ind (middle) and EMMI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.

3.54 The final run of the five sequential designs for the DGP problem with an initial design of size 10 and 10 points added sequentially via EHI1-Ind (top), EWH1-Ind (middle) and KEI-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
3.55 The final run of the five sequential designs for the DGP problem with an initial design of size 10 and 10 points added sequentially via EHI2-Ind (top), EWHI2-Ind (middle) and KEI1-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. ................................................................. 177

3.56 The final run of the five sequential designs for the DGP problem with an initial design of size 10 and 10 points added sequentially via PI-Ind (top), ECI1-Ind (middle) and ECI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. ................................................................. 178

3.57 The final run of the five sequential designs for the DGP problem with an initial design of size 10 and 10 points added sequentially via EECI1-Ind (top) and EECI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. ................................................................. 179

3.58 The final run of the five sequential designs for the DGP problem with an initial design of size 10 and 10 points added sequentially via EMMI-Dep (top), EMAX1-Dep (middle) and EMMI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. ................................................................. 180

3.59 The final run of the five sequential designs for the DGP problem with an initial design of size 10 and 10 points added sequentially via EHI1-Dep (top), EWHI1-Dep (middle) and KEI-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. ................................................................. 181

3.60 The final run of the five sequential designs for the DGP problem with an initial design of size 10 and 10 points added sequentially via EHI2-Dep (top), EWHI2-Dep (middle) and KEI1-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. ................................................................. 182
3.61 The final run of the five sequential designs for the DGP problem with an initial design of size 10 and 10 points added sequentially via PI-Dep (top) and ECI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. 183

3.62 The final run of the five sequential designs for the DGP problem with an initial design of size 10 and 10 points added sequentially via EECI1-Dep (top) and EECI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. 184

3.63 The final run of the five sequential designs for the DGP problem with an initial design of size 20 and 10 points added sequentially via EMMI-Ind (top), EMAX1-Ind (middle) and EMMI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. 185

3.64 The final run of the five sequential designs for the DGP problem with an initial design of size 20 and 10 points added sequentially via EHI1-Ind (top), EWHI1-Ind (middle) and KEI-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. 186

3.65 The final run of the five sequential designs for the DGP problem with an initial design of size 20 and 10 points added sequentially via EHI2-Ind (top), EWHI2-Ind (middle) and KEI1-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. 187

3.66 The final run of the five sequential designs for the DGP problem with an initial design of size 20 and 10 points added sequentially via PI-Ind (top), ECI1-Ind (middle) and ECI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points. 188
3.67 The final run of the five sequential designs for the DGP problem with an initial design of size 20 and 10 points added sequentially via EECI1-Ind (top) and EECI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.

3.68 The final run of the five sequential designs for the DGP problem with an initial design of size 20 and 10 points added sequentially via EMMI-Dep (top), EMAX1-Dep (middle) and EMMI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.

3.69 The final run of the five sequential designs for the DGP problem with an initial design of size 20 and 10 points added sequentially via EHI1-Dep (top), EWHI1-Dep (middle) and KEI-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.

3.70 The final run of the five sequential designs for the DGP problem with an initial design of size 20 and 10 points added sequentially via EHI2-Dep (top), EWHI2-Dep (middle) and KEI1-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.

3.71 The final run of the five sequential designs for the DGP problem with an initial design of size 20 and 10 points added sequentially via PI-Dep (top) and ECI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.

3.72 The final run of the five sequential designs for the DGP problem with an initial design of size 20 and 10 points added sequentially via EECI1-Dep (top) and EECI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
3.73 Boxplots of the component sums for each input in the Pareto set approximation created by the expected improvement criteria indicated on the horizontal axis for the ZLT1 problem. Component sums close to 1 are indicative of inputs near the true Pareto set. ................. 202

3.74 Diagram of the Kodak disposable camera. The diagonals of interest (a3 and b3) can be seen in the top right. ......................... 204

3.75 Sequentially added inputs for the Kodak disposable camera problem using EMMI-Ind. The original inputs are magenta circles labeled 1-9, while the sequentially added inputs are black crosses numbered 10-20, with 10 being the first sequentially added point, and 20 being the last. ................................................. 206

3.76 Sequentially added outputs for the Kodak disposable camera problem using EMMI-Ind. The original outputs are magenta circles labeled 1-9, while the sequentially added outputs are black crosses numbered 10-20, with 10 being the first sequentially added output, and 20 being the last. ................................................. 206

3.77 Sequentially added inputs for the Kodak disposable camera problem using ECI2-Ind. The original inputs are magenta circles labeled 1-9, while the sequentially added inputs are black crosses numbered 10-20, with 10 being the first sequentially added point, and 20 being the last. ................................................. 208

3.78 Sequentially added outputs for the Kodak disposable camera problem using ECI2-Ind. The original outputs are magenta circles labeled 1-9, while the sequentially added outputs are black crosses numbered 10-20, with 10 being the first sequentially added output, and 20 being the last. ................................................. 208

3.79 Feasible \( y_1(\cdot) \) (cross-sectional area in mm\(^2\)) ..................... 211

3.80 Feasible \( y_2(\cdot) \) (beam bending stress in MPa). ..................... 212

3.81 Dark black circles are the approximated true Pareto front for the Nowacki beam problem ................................. 213

3.82 Dark black circles are the approximated true Pareto set for the Nowacki beam problem ................................. 213
3.83 Comparison of Pareto front approximations using either scaling as defined in (3.2.2) or (3.2.1). Estimated Pareto fronts using EHI1 with (3.2.2) so the observed constrained maximums are the reference point (top left), EHI1 with (3.2.1) (top right) so the observed unconstrained maximums are the reference point, EMMI with (3.2.2) (middle left), EMMI with (3.2.1) (middle right), KEI with (3.2.2) (bottom left), and KEI with (3.2.1) (bottom right).

3.84 Estimated Pareto sets (left) and estimated Pareto fronts (right) using EMMI (top), EMAX1 (middle) and EMMI2 (bottom). The blue line in each plot represent the true Pareto set/front.

3.85 Estimated Pareto sets (left) and estimated Pareto fronts (right) using EHI1 (top), EWHI1 (middle) and KEI (bottom). The blue line in each plot represent the true Pareto set/front.

3.86 Estimated Pareto sets (left) and estimated Pareto fronts (right) using EHI2 (top), EWHI2 (middle) and KEI1 (bottom). The blue line in each plot represent the true Pareto set/front.

3.87 Estimated Pareto sets (left) and estimated Pareto fronts (right) using PI (top), ECI1 (middle) and ECI2 (bottom). The blue line in each plot represent the true Pareto set/front.

3.88 Estimated Pareto sets (left) and estimated Pareto fronts using EECI1 (top) and EECI2 (bottom). The blue line in each plot represent the true Pareto set/front.

4.1 Prediction of 4.1.4 based on the initial 7-point design using a constant mean GP (left) and a GP with a quadratic mean $\beta_0 + \beta_1 x + \beta_2 x^2$ (right). The dashed line is the predicted surface, and the solid line is the true surface.

4.2 10-point maximin Latin hypercube
4.3  *Top Left:* Plot of $y(x_1, x_2)$ in (4.1.5) over $[0, 1]^2$. *Bottom Left:* Predictions using a constant mean model based on 10-point design. *Bottom Right:* Prediction using a GP with mean 4.1.6 based on 10-point design. *Top Right:* Squared prediction errors of the quadratic mean model subtracted from the squared prediction errors in the constant mean model computer over a $50 \times 50$ grid in $[0, 1]^2$. . . . . . . . . . . 233
CHAPTER 1

INTRODUCTION

The goal of this chapter is to give a broad overview of the main topics considered in this thesis, motivate the methodology presented in subsequent chapters, and to introduce some of the statistical models and numerical tools that will be used throughout the thesis.

1.1 Computer Experiments

Computer experiments use computer simulators to study the input-output relationship between response variables and explanatory variables instead of physical experimentation. They are feasible when scientific theory exists to relate the explanatory variables to the response, and sufficient numerical methods and computing power exist to implement the scientific theory. Computer experiments are useful when physical experiments, the gold standard for establishing a cause and effect relationship, are impossible due to practical, economical, or ethical concerns.

Computer experiments have several unique features. They are typically deterministic, so several key components of physical experiments, such as randomizing the order of treatments, blocking, and replication, are not necessary. Computer experiments are typically biased representations of the true input-output relationship.
Additionally, computer experiments are essentially black-box functions; by black-box, we mean that the functional form of the relationship between the inputs and the response is unknown, even though the response can be computed for any given input. Thus, the source of uncertainty is the unknown form of the relationship between inputs and outputs, not random error. Finally, computer experiments often have long run times, as it can take several hours or even days to obtain a response for a single setting of the input variables.

The design of a computer experiment is the set of inputs at which we compute the outputs. When all of the inputs are quantitative (which we will assume throughout this thesis), there are typically two ways in which one selects a design. With a fixed design, all design points are chosen prior to performing the computer experiment. In this situation, since the computer code is a black-box, one generally believes interesting features of the functional relationship between the inputs and the outputs are just as likely to be in one region as another. Therefore, the goal is spread the inputs evenly across the input space. Such designs are said to be “space-filling” or “exploratory”. While there are several ways to define the notion of space-filling and create designs that meet these criteria, this thesis will mainly use maximin Latin hypercube designs (LHDs) (see Chapter 5, Santner et al. (2003)) when space-filling designs are necessary.

As opposed to fixed designs, one can choose the inputs via criteria-based designs. In this situation, one typically starts with a fixed space-filling design, fits an interpolating process model based on the observed outputs and the initial design, and then uses the fitted model and a specially developed criterion (consistent with the experimental goals) to determine the inputs at which to observe the computer code
next. The process is repeated until the computational budget has been exhausted, or a stopping criterion has been attained.

1.2 Motivating Examples

In this section, the two main topics of this thesis, multiobjective optimization and sensitivity analysis of computer experiments, will be introduced via two examples from the literature.

1.2.1 Design of a Piezoelectric Bimorph Actuator

The first example is an engineering design problem described in Wilson et al. (2001). In this problem, the goal is to design a piezoelectric bimorph actuator for minimally invasive surgery. The actuator is created by laminating several thin strips of piezoelectric ceramic material to create two separate beams of sandwiched material. The beams are then cantilevered, with one beam over the other. When opposing voltages are run through the two beams, they are bent toward each other in a simple grasping motion, not unlike a finger and a thumb, that can be used to hold suture needles. The engineers are faced with two competing design objectives. The beams must have enough tip deflection so that the grasper can close completely, and the beams must be strong enough to provide sufficient tip force to prevent the needle from rolling or falling out from between the beams. Thus, the goal is to find some sort of optimal trade-off between strength and flexibility. The design variables are the thickness of the beam at five different locations along the beam.

The engineers used a finite-element model to compute tip force and tip deflection for a given set of thickness values. Unfortunately, the finite element model was time-consuming to evaluate, and many existing algorithms designed to find inputs that
offer an optimal trade-off between the two objectives require more function evaluations than allowed by the computational budget. Therefore, this problem requires methodology that allows one to find inputs that offer an optimal trade-off in the outputs with as few runs of the computational model as possible.

Such methodology will be developed in the sequential criteria-based design of computer experiments framework. In Chapter 2, sequential design criteria consistent with the goal of multiobjective optimization (i.e., finding the optimal trade-offs) will be developed. Chapter 3 will use these criteria on several test functions and real-world applications.

1.2.2 Oil-Field Simulator

The next example is described in Oakley and O’Hagan (2004). Here, a computer model predicts pressures at various time points for a set of wells in a hydrocarbon reservoir; the authors focus only on one particular well at one time point. The inputs for the code include 40 total quantitative variables, where 7 measure permeability, and 33 measure fault transmissibility. The goal is to determine which inputs have the largest influence on the pressure, and which inputs have a negligible effect on pressure. This study of the variability of the output of a computer experiment with respect to changes with the inputs is referred to as sensitivity analysis, and one way to perform such a study involves the calculation of sensitivity indices, which are introduced in detail in Section 1.7.

However, current methodology and software for estimating sensitivity indices in computer experiments is quite limited. Most methods make restrictive assumptions that do not allow one to account for large scale polynomial trends in the computer
code, and most are based on modeling assumptions that are either computationally inefficient or infeasible for very large designs. A goal of this thesis is to develop methodology that allows one to estimate sensitivity indices in situations where current methodology and its current implementations fail or are inefficient. This research is presented in Chapter 4.

1.3 Modeling Multiple-Output Computer Experiments

A Bayesian viewpoint will be adopted when modeling the uncertainty regarding the true input-output relationship in multiple-output computer code. We denote the output of a computer experiment as an $m$-dimensional function $y(\cdot) = (y_1(\cdot), \ldots, y_m(\cdot))^\top$ from a $d$-dimensional input space $\mathcal{X} \subset \mathbb{R}^d$ with positive volume in the $d^{th}$ dimension. We assume $y(\cdot)$ is a realization of an $m$-variate Gaussian process $Y(\cdot)$. Specifically,

$$Y(x) = F(x)\beta + W(x)$$

(1.3.1)

where

$$F(x) = \begin{pmatrix}
    f_1^\top(x) & 0_{1 \times p_2} & \cdots & 0_{1 \times p_m} \\
    0_{1 \times p_1} & f_2^\top(x) & \cdots & 0_{1 \times p_m} \\
    \vdots & \vdots & \ddots & \vdots \\
    0_{1 \times p_1} & 0_{1 \times p_2} & \cdots & f_m^\top(x)
\end{pmatrix}$$

and

$$f_i^\top(x) = (f_{i,1}(x)), f_{i,2}(x)), \ldots, f_{i,p_i}(x)).$$

These $m$ regression functions are known. The vector $\beta$ is partitioned as $\beta = (\beta_1^\top, \ldots, \beta_m^\top)^\top$ where $\beta_i = (\beta_{i,1}, \ldots, \beta_{i,p_i})^\top$ for each $i = 1, \ldots, m$. The $\beta_i$ are the unknown regression coefficients for the $i^{th}$ function, $1 \leq i \leq m$. The total number of coefficients is denoted $p = \sum_{i=1}^m p_i$. We assume that $W(x)$ is an $m$-variate Gaussian
process with zero mean and covariance structure specified by the $m \times m$ matrix-valued covariance function $K(\cdot, \cdot)$ which depends on a vector of parameters $\psi$. This means that for any $x, x' \in \mathcal{X}$, we have $\text{Cov} (Y(x), Y(x')) = K(x, x')$ and, componentwise, $\text{Cov} (Y_i(x), Y_j(x')) = [K(x, x')]_{i,j}$. Additionally, we will assume the usual noninformative prior on $\beta$, i.e., $\pi(\beta) \propto 1$.

### 1.3.1 Predictive Distribution

Suppose we have evaluated $y(\cdot)$ at an initial design $D_n = \{x_1, \ldots, x_n\} \subset \mathcal{X}$. Let $y^{m,n} = (y^T(x_1), \ldots, y^T(x_n))^T$ be the vector of outputs at $D_n$. We wish to predict $y(\cdot)$ at inputs $(x_1^0, \ldots, x_n^0)$. Letting $Y^0 = (Y^T(x_1^0), \ldots, Y^T(x_n^0))^T$, our goal in this subsection is to derive the posterior predictive distribution of $[Y^0 | Y^{m,n} = y^{m,n}]$ assuming the covariance parameters $\psi$ are known.

Based on the Gaussian assumptions, we have

$$
\begin{bmatrix}
Y^0 \\
y_{m,n}
\end{bmatrix} \mid \beta \sim N \left( \begin{bmatrix}
F_0 \\
F_n
\end{bmatrix} \beta, \begin{bmatrix}
\Sigma^0 & \Sigma_{0,0,n} \\
\Sigma_{0,n,0} & \Sigma_{m,n}
\end{bmatrix} \right)
$$

where $F_0 = [F^T(x_1^0), \ldots, F^T(x_n^0)]^T$, $F_n = [F^T(x_1), \ldots, F^T(x_n)]^T$, $\Sigma_0 = K(x, x)$,

$$
\Sigma^0 = 
\begin{bmatrix}
\Sigma_0 & K(x_1^0, x_2^0) & \ldots & K(x_1^0, x_{n_0}^0) \\
K(x_1^0, x_2^0) & \Sigma_0 & \ldots & K(x_2^0, x_{n_0}^0) \\
\vdots & \ddots & \ddots & \vdots \\
K(x_1^0, x_{n_0}^0) & K(x_2^0, x_{n_0}^0) & \ldots & \Sigma_0
\end{bmatrix},
\tag{1.3.2}
$$

$$
\Sigma_{m,n} = 
\begin{bmatrix}
\Sigma_0 & K(x_1, x_2) & \ldots & K(x_1, x_n) \\
K(x_1, x_2) & \Sigma_0 & \ldots & K(x_2, x_n) \\
\vdots & \ddots & \ddots & \vdots \\
K(x_1, x_n) & K(x_2, x_n) & \ldots & \Sigma_0
\end{bmatrix},
\tag{1.3.3}
$$

and

$$
\Sigma_{0,0,n} = 
\begin{bmatrix}
K(x_1^0, x_1) & K(x_1^0, x_2) & \ldots & K(x_1^0, x_{n_0}) \\
K(x_2^0, x_1) & K(x_2^0, x_2) & \ldots & K(x_2^0, x_{n_0}) \\
\vdots & \ddots & \ddots & \vdots \\
K(x_{n_0}^0, x_1) & K(x_{n_0}^0, x_2) & \ldots & K(x_{n_0}^0, x_n)
\end{bmatrix}.
\tag{1.3.4}
$$
We can derive the distribution of \([Y^0] \mid Y^{m,n} = y^{m,n}\) with the following result:

**Theorem 1.** Let \(Y_1\) be an \(n_1 \times 1\) random vector and \(Y_2\) be an \(n_2 \times 1\) random vector with

\[
\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} \mid \beta \sim N \left( \begin{bmatrix} \mathcal{F}_1 \\ \mathcal{F}_2 \end{bmatrix} \beta, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{12}^\top & \Sigma_{22} \end{bmatrix} \right),
\]

where \(\beta\) is a \(p \times 1\) vector, \(\mathcal{F}_1\) and \(\mathcal{F}_2\) are \(n_1 \times p\) and \(n_1 \times p\) matrices respectively with full column rank, \(\Sigma_{11} = \text{cov} \left( Y_1, Y_1^\top \right) \), \(\Sigma_{22} = \text{cov} \left( Y_2, Y_2^\top \right)\), and \(\Sigma_{12} = \text{cov} \left( Y_1, Y_2^\top \right)\).

Also, assume that \(\beta\) has an improper prior \(f(\beta) \propto 1\). Then, \([Y_1 \mid Y_2 = y_2] \sim N(u_{1|2}, \Sigma_{1|2})\) where

\[
\begin{align*}
u_{1|2} &= \mathcal{F}_1 \hat{\beta} + \Sigma_{12} \Sigma_{22} (y_2 - \mathcal{F}_2 \hat{\beta}), \\
\Sigma_{1|2} &= \Sigma_{11} - \Sigma_{12} \Sigma_{22}^{-1} \Sigma_{12}^\top \\
&+ (\mathcal{F}_1 - \Sigma_{12} \Sigma_{22}^{-1} \mathcal{F}_2) (\mathcal{F}_2^\top \Sigma_{22}^{-1} \mathcal{F}_2)^{-1} (\mathcal{F}_1 - \Sigma_{12} \Sigma_{22}^{-1} \mathcal{F}_2)^\top
\end{align*}
\]

and

\[
\hat{\beta} = (\mathcal{F}_2^\top \Sigma_{22}^{-1} \mathcal{F}_2)^{-1} \mathcal{F}_2^\top \Sigma_{22}^{-1} y_2.
\]

**Proof.** See Appendix A. \(\square\)

By a straightforward application of Theorem 1, \([Y^0] \mid Y^{m,n} = y^{m,n}\) \(\sim N(\hat{y}^0, S^0)\) where

\[
\begin{align*}
\hat{y}^0 &= \mathcal{F}_0 \hat{\beta} + \Sigma_{0,n_0,n} \Sigma_{m,n}^{-1} (y^{m,n} - \mathcal{F}_n \hat{\beta}), \\
S^0 &= \Sigma^0 - \Sigma_{0,n_0,n} \Sigma_{m,n}^{-1} \Sigma_{0,n_0,n}^\top \\
&+ (\mathcal{F}_0 - \Sigma_{0,n_0,n} \Sigma_{m,n}^{-1} \mathcal{F}_n) (\mathcal{F}_n^\top \Sigma_{m,n}^{-1} \mathcal{F}_n)^{-1} (\mathcal{F}_0 - \Sigma_{0,n_0,n} \Sigma_{m,n}^{-1} \mathcal{F}_n)^\top
\end{align*}
\]

and

\[
\hat{\beta} = (\mathcal{F}_n^\top \Sigma_{m,n}^{-1} \mathcal{F}_n)^{-1} \mathcal{F}_n^\top \Sigma_{m,n}^{-1} y^{m,n}.
\]
We can partition $\hat{y}^0$ into $n_0 m \times 1$ vectors, so that

$$
\hat{y}^0 = \left( E \left[ Y(x_1^0) \right] Y^{m,n} \right)^\top, \ldots, E \left[ Y(x_{n_0}^0) \right] Y^{m,n} \right)^\top
$$

= \left( \hat{y}^\top (x_1^0), \ldots, \hat{y}^\top (x_{n_0}^0) \right)^\top
$$

(1.3.11)

and

$$
S^0 = \begin{bmatrix}
S(x_1^0) & S(x_1^0, x_2^0) & \cdots & S(x_1^0, x_{n_0}^0) \\
S(x_1^0, x_2^0) & S(x_2^0) & \cdots & S(x_2^0, x_{n_0}^0) \\
\vdots & \vdots & \ddots & \vdots \\
S(x_1^0, x_{n_0}^0) & S(x_2^0, x_{n_0}^0) & \cdots & S(x_{n_0}^0)
\end{bmatrix},
$$

(1.3.12)

where $S(x_i^0) = \text{cov} (Y(x_i^0), Y(x_i^0)|Y^{m,n})$ and $S(x_i^0, x_j^0) = \text{cov} (Y(x_i^0), Y(x_j^0)|Y^{m,n})$.

If we make the assumption that suppose that $Y_1(\cdot), \ldots, Y_m(\cdot)$ are independent, we can prove the following result:

**Theorem 2.** Consider the $m$-variate Gaussian process model (1.3.1). Suppose that $W(x) = (W_1(x), \ldots, W_m(x))$ where $W_1(x), \ldots, W_m(x)$ are independent stationary GaSPs with covariances $\sigma_i^2 R_i(\cdot; \theta_i)$, where $\sigma_i^2$ and $R_i(\cdot; \theta_i)$ are the variance and correlation of the $i$th process, $1 \leq i \leq m$, and $\theta_i$ is a vector of correlation parameters. Let $Y_i = (Y_i(x_1^0), \ldots, Y_i(x_{n_0}^0))$ and $y_i^n = (y_i(x_1), \ldots, y_i(x_n))$ for $1 \leq i \leq m$. Also, let $F_i^n = [f_i(x_1) \cdots f_i(x_n)]^\top$ and $F_i^0 = [f_i(x_1^0) \cdots f_i(x_{n_0}^0)]^\top$. Additionally, let $R_k$ be an $n \times n$ matrix with $[R_k]_{ij} = R_k (x_i - x_j; \theta_k)$ and $r_k$ be an $n \times n_0$ matrix with $r_k = [r_k(x_1^0), \ldots, r_k(x_{n}^0)]$, where each

$$
r_k(x_i^0) = \left[ R_k (x_1 - x_i^0; \theta_k), \ldots, R_k (x_n - x_i^0; \theta_k) \right]^\top.
$$

Then, $[Y_i^0 | Y^{m,n} = y^{m,n}] \sim N \left( \hat{y}_i^0, \sigma_i^2 R_i^0 \right), 1 \leq i \leq m$, where

$$
\hat{y}_i^0 = F_i^0 \hat{\beta}_i + r_i^\top R_i^{-1} \left( y_i^n - F_i^n \hat{\beta}_i \right),
$$

(1.3.13)
\[ R_{i}^{0} = I_{n} - r_{i}^{\top} R_{i}^{-1} r_{i} \]
\[ + (F_{i}^{0} - r_{i}^{\top} R_{i}^{-1} F_{i}^{n})(F_{i}^{n} R_{i}^{-1} F_{i}^{n})^{-1}(F_{i}^{0} - r_{i}^{\top} R_{i}^{-1} F_{i}^{n})^{\top}, \]  \hspace{1cm} (1.3.14) 

and

\[ \hat{\beta}_{i} = (F_{i}^{0} - r_{i}^{\top} R_{i}^{-1} F_{i}^{n})(F_{i}^{n} R_{i}^{-1} F_{i}^{m})^{-1} F_{i}^{n} R_{i}^{-1} y_{i}^{n}. \]  \hspace{1cm} (1.3.15) 

Additionally, for \( i \neq j \), \([Y_{i}^{0}|Y_{m,n}^{m,n} = y_{m,n}] \perp [Y_{j}^{0}|Y_{m,n}^{m,n} = y_{m,n}]\).

**Proof.** See Appendix A. \(\square\)

While \( n_{0} \) can be any positive integer, the cases of \( n_{0} = 1 \) and \( n_{0} = 2 \) are of particular interest in subsequent chapters, so these cases are now discussed in detail.

**Special Case: \( n_{0} = 1 \)**

In this particular case, we are interested in obtaining a single prediction at one particular input \( x_{0} \). Here \([Y(x_{0})|Y_{m,n}^{m,n} = y_{m,n}] \sim N(\hat{y}(x_{0}), S(x_{0}))\), where

\[ \hat{y}(x_{0}) = F(x_{0}) \hat{\beta} + \Sigma_{0,n} \Sigma_{m,n} (y_{m,n} - F_{n} \hat{\beta}), \]  \hspace{1cm} (1.3.16) 

\[ S(x_{0}) = \Sigma_{0} - \Sigma_{0,n} \Sigma_{m,n} \Sigma_{m,n}^{\top} \]
\[ + (F(x_{0}) - \Sigma_{0,n} \Sigma_{m,n} F_{n})(F_{n} \Sigma_{m,n} F_{n})^{-1}(F(x_{0}) \]
\[ - \Sigma_{0,n} \Sigma_{m,n} F_{n})^{\top}, \]  \hspace{1cm} (1.3.17) 

and

\[ \Sigma_{0,n} = \text{cov} (Y(x_{0}), Y_{m,n}^{m,n}) = [K(x_{0}, x_{1}), K(x_{0}, x_{2}), \ldots, K(x_{0}, x_{n})]. \]

There is a connection between the predictive distribution \([Y(x_{0})|Y_{m,n}^{m,n} = y_{m,n}]\) when \( n_{0} = 1 \) and the optimal linear predictor of \( y(x_{0}) \). The expressions \( \hat{y}(x_{0}) \) and \( S(x_{0}) \) are equivalent to the best linear unbiased predictor (BLUP) of \( y(x_{0}) \) and the associated
mean square prediction error (MSPE) matrix, respectively. The details are discussed in Appendix A.3, including a formal definition the BLUP and MSPE matrix in the multivariate setting.

**Special Case: \( n_0 = 2 \) with Independent Output**

Suppose the outputs are independent. Then, given \( x \) and \( u \) in \( \mathcal{X} \),

\[
Y_*^0 = (Y_1(x), Y_1(u), \ldots, Y_m(x), Y_m(u)),
\]

and we can partition each \( \hat{y}_i^0 \) as

\[
\hat{y}_i^0 = \begin{bmatrix}
\hat{y}_i(x) \\
\hat{y}_i(u)
\end{bmatrix}
\]

(1.3.18)

Notice that \( \hat{y}_i(x) = E[Y_i(x)|Y_{m,n} = y_{m,n}] \) and \( \hat{y}_i(u) = E[Y_i(u)|Y_{m,n} = y_{m,n}] \). We can partition each \( R_i^0 \) from (1.3.14) as

\[
R_i^0 = \begin{bmatrix}
r_i(x) & r_i(x, u) \\
r_i(x, u) & r_i(u)
\end{bmatrix}
\]

(1.3.19)

where

\[
r_i(x) \equiv \frac{1}{\sigma_i^2} \text{cov} (Y_i(x), Y_i(x)|Y_{m,n} = y_{m,n}) \quad (1.3.20)
\]

\[
r_i(u) \equiv \frac{1}{\sigma_i^2} \text{cov} (Y_i(u), Y_i(u)|Y_{m,n} = y_{m,n}) \quad (1.3.21)
\]

\[
r_i(x, u) \equiv \frac{1}{\sigma_i^2} \text{cov} (Y_i(x), Y_i(u)|Y_{m,n} = y_{m,n}). \quad (1.3.22)
\]

**Special Case: \( n_0 = 1 \) and \( m = 1 \)**

Here, we wish to predict computer code with only one output at a single input \( x^0 \). In this case, we will denote the computer code as \( y(\cdot) \) (rather than \( y(\cdot) \)), and our output at \( D_n \) will be denoted \( y^n = (y(x_1), \ldots, y(x_n))^\top \). Then, (1.3.1) becomes

\[
Y(x) = f^\top(x)\beta + W(x),
\]

(1.3.23)
where $\beta$ is a $p \times 1$ vector of unknown regression parameters,

$$f^\top(x) = (f_1(x), f_2(x), \ldots, f_p(x))$$

is a vector of $p$ known regression function, and $W(\cdot)$ is a univariate stationary Gaussian process with covariance function $\text{Cov}(x, x') = K(x - x')$. Assuming a product correlation structure

$$K(x - x') = \sigma^2 R(x, x'; \theta) \quad (1.3.24)$$

(where $\theta$ is a vector of correlation parameters) and letting

$$F = [f(x_1) \cdots f(x_n)]^\top,$$
$$r = [R(x^0 - x_1; \theta), \ldots, R(x^0 - x_n; \theta)]$$

and $R$ be an $n \times n$ matrix with $(i, j)^{th}$ element equal to $R(x_i - x_j; \theta)$, we have $[y(x_0) | Y^n = y^n] \sim N(\hat{y}(x), s^2(x))$, assuming $\pi(\beta) \propto 1$. Here,

$$\hat{y}(x) = f^\top(x^0)\hat{\beta} + r^\top R_i^{-1} \left( y^n - F\hat{\beta} \right) \quad (1.3.25)$$

where

$$\hat{\beta} = (F^T R^{-1} F)^{-1} F^T R^{-1} y^n, \quad (1.3.26)$$

and

$$s^2(x) = \sigma^2 \left\{ 1 - r^\top R^{-1} r \right. $$
$$+ \left. (F - r^\top R^{-1} F)(F^T R^{-1} F)^{-1}(F - r^\top R^{-1} F)^\top \right\}. \quad (1.3.27)$$

Just as in the multiple-output case, (1.3.26) and (1.3.27) are equal to the best linear unbiased predictor (BLUP) of $y(x)$ and the mean square prediction error (MSPE) of
the BLUP, respectively. (The BLUP of \( y(x) \) is the \( \bar{y}(x) \) that minimizes the mean square prediction error \( E \left[ (y(x) - \bar{y}(x))^2 \right] \) subject to \( E[\bar{y}(x)] = f^\top(x)\beta \) and \( \bar{y}(x) = b'y_n \), where \( b \) is an \( n \times 1 \) vector.)

### 1.3.2 Covariance Structure

Some methods that one can use to model

\[
\text{Cov}(Y(x), Y(x')) = K(x, x')
\]

will now be discussed. This thesis will mainly focus on the independence model and the nonseparable linear model of coregionalization. However, some other ways one could emulate multiple output computer code will also be briefly reviewed.

**Independence Model**

Under the assumption that the processes \( Y_1(\cdot), \ldots, Y_m(\cdot) \) are independent with known covariances of the form \( \sigma_i^2 R_i(\cdot; \theta_i) \), Theorem 2 details the predictive distribution of each component conditional on \( y^n_i = (y_i(x_1), \ldots, y_i(x_n)) \) for \( i = 1, \ldots, m \). Posterior means and covariances for each component can be computed via (1.3.13)-(1.3.15).

When each \( \sigma_i^2 \) and \( \theta_i \) are unknown, we can estimate these unknown quantities using maximum likelihood (ML) or restricted maximum likelihood (REML) and plug their estimates into (1.3.13)-(1.3.15). One can show (see Santner et al. (2003)) that MLEs of each \( \sigma_i^2 \) and \( \theta_i \) are

\[
\hat{\sigma}_i^2 = \frac{1}{n} \left( y_i^n - F_i^n \hat{\beta}_i \right)^T R_i^{-1} \left( y_i^n - F_i^n \hat{\beta}_i \right)
\]

and

\[
\hat{\theta}_i = \text{argmax} \left\{ -n \log(\hat{\sigma}_i^2) - \log(|R_i|) \right\}.
\]
Similarly, one can show that the REML estimates of each $\sigma_i^2$ and $\theta_i$ are

$$\tilde{\sigma}_i^2 = \frac{n}{n - p_i} \hat{\sigma}_i^2 = \frac{1}{n - p_i} \left( y_i^n - F_i^n \hat{\beta}_i \right)^T R_i^{-1} \left( y_i^n - F_i^n \hat{\beta}_i \right)$$

and

$$\tilde{\theta}_i = \arg \max \left\{ -(n - p_i) \log \left( \hat{\sigma}_i^2 \right) - \log(|R_i|) \right\}.$$

**Nonseparable Linear Model of Coregionalization**

The *nonseparable linear model of coregionalization* (NLMC) is used in the geostatistics literature in Gelfand et al. (2004), Banerjee and Johnson (2006) and Banerjee et al. (2008), and in the computer experiments literature in Fricker et al. (2010).

In this model, $Y(x)$ is assumed to be the process of the form

$$Y(x) = F(x)\beta + AZ(x) \quad (1.3.28)$$

where $A$ is a symmetric $m \times m$ positive-definite matrix,

$$\beta = (\beta_1 \ldots \beta_m)^T, \quad (1.3.29)$$

and $Z(x) = (Z_1(x), \ldots, Z_m(x))^T$ is an $m \times 1$ vector of mutually independent stationary Gaussian processes with *zero mean* and *unit variance*. Thus, in (1.3.1), we are assuming $W(x) = AZ(x)$. The mean of the process $Y(x)$ is $F(x)\beta$; its between-output covariance is determined by $A$.

In more detail, it is assumed that the process $Z_i(x)$ has zero mean, unit variance, and a correlation function of the form $R_i(x - x'; \theta_i)$, where $\theta_i$ is a vector of correlation parameters. For example, one such choice for the correlation function is the Gaussian correlation function

$$R_i(x - x'; \theta_i) = \exp \left\{ \sum_{j=1}^d \theta_{i,j} (x_i - x'_j)^2 \right\}. \quad (1.3.30)$$
Under these assumptions it is straightforward to show that

\[
\text{Cov}(Y(x), Y(x')) = \mathbf{A} \text{diag}(R(x - x'; \theta_1), \ldots, R(x - x'; \theta_m)) \mathbf{A}^\top,
\]  

(1.3.31)

so that, when \(x = x'\),

\[
\text{Cov}(Y(x), Y(x)) = \mathbf{A} \mathbf{A}^\top = \mathbf{A} \mathbf{A} \equiv \Sigma_0.
\]  

(1.3.32)

The matrix \(\mathbf{A}\) is to be thought of as the unique matrix square root, defined via eigen decomposition, of the cross-covariance matrix \(\Sigma_0\). Thus the model states that the association between the components of \(Y(x)\) is the same for all \(x\) and that the component process \(Y_i(x)\) is stationary with variance \(\sum_{j=1}^m a_{ij}^2\). It is easy to compute that \(\Sigma_{m,n}\), as defined in (1.3.3), is equal to

\[
\begin{pmatrix}
\sum_0 & \text{Cov}(Y(x_1), Y(x_2)) & \cdots & \text{Cov}(Y(x), Y(x_n)) \\
\text{Cov}(Y(x_1), Y(x_2)) & \sum_0 & \cdots & \text{Cov}(Y(x_2), Y(x_n)) \\
\vdots & \vdots & \ddots & \vdots \\
\text{Cov}(Y(x_1), Y(x_n)) & \text{Cov}(Y(x_2), Y(x_n)) & \cdots & \sum_0
\end{pmatrix}
\]

\[
= \begin{pmatrix}
\mathbf{A} \mathbf{A} & \mathbf{A} \Lambda_{1,2} \mathbf{A} & \cdots & \mathbf{A} \Lambda_{1,n} \mathbf{A} \\
\mathbf{A} \Lambda_{1,2} \mathbf{A} & \mathbf{A} \mathbf{A} & \cdots & \mathbf{A} \Lambda_{2,n} \mathbf{A} \\
\vdots & \vdots & \ddots & \vdots \\
\mathbf{A} \Lambda_{1,n} \mathbf{A} & \mathbf{A} \Lambda_{2,n} \mathbf{A} & \cdots & \mathbf{A} \mathbf{A}
\end{pmatrix},
\]  

(1.3.33)

where \(\Lambda_{i,j} = \oplus_{k=1}^m R_k(x_i - x_j; \theta_k)\) and \(\oplus\) is the usual direct sum operator. Similarly, \(\Sigma^0\), as defined in (1.3.2), is equal to

\[
\begin{pmatrix}
\mathbf{A} \mathbf{A} & \mathbf{A} \Lambda_{0,1,2}^0 \mathbf{A} & \cdots & \mathbf{A} \Lambda_{0,1,n}^0 \mathbf{A} \\
\mathbf{A} \Lambda_{0,1,2}^0 \mathbf{A} & \mathbf{A} \mathbf{A} & \cdots & \mathbf{A} \Lambda_{0,2,n}^0 \mathbf{A} \\
\vdots & \vdots & \ddots & \vdots \\
\mathbf{A} \Lambda_{0,1,n}^0 \mathbf{A} & \mathbf{A} \Lambda_{0,2,n}^0 \mathbf{A} & \cdots & \mathbf{A} \mathbf{A}
\end{pmatrix},
\]  

(1.3.34)

where \(\Lambda_{0,i,j}^0 = \oplus_{k=1}^m R_k(x_i^0 - x_j^0; \theta_k)\), and \(\Sigma_{0,n,n}\), as defined in (1.3.4), is equal to

\[
\begin{pmatrix}
\mathbf{A} \Lambda_{0,1,1}^0 \mathbf{A} & \mathbf{A} \Lambda_{0,1,2}^0 \mathbf{A} & \cdots & \mathbf{A} \Lambda_{0,1,n}^0 \mathbf{A} \\
\mathbf{A} \Lambda_{0,2,1}^0 \mathbf{A} & \mathbf{A} \Lambda_{0,2,2}^0 \mathbf{A} & \cdots & \mathbf{A} \Lambda_{0,2,n}^0 \mathbf{A} \\
\vdots & \vdots & \ddots & \vdots \\
\mathbf{A} \Lambda_{0,n,1}^0 \mathbf{A} & \mathbf{A} \Lambda_{0,n,2}^0 \mathbf{A} & \cdots & \mathbf{A} \Lambda_{0,n,n}^0 \mathbf{A}
\end{pmatrix},
\]  

(1.3.35)
\[
\Lambda_{0,i,j} = \oplus_{k=1}^{m} R_k(x_i^0 - x_j; \theta_k).
\]

When \( A \) and \( \theta \) are unknown, we can estimate them using maximum likelihood (ML) or restricted maximum likelihood (REML) and their estimates are plugged into (1.3.8)-(1.3.10). Specifically, the ML estimates of \( A \) and \( \theta \) are

\[
\left( \hat{A}, \hat{\theta} \right) = \arg\max \left\{ -\frac{1}{2} \log(|\Sigma_{m,n}|) - \frac{1}{2} \left( y_{m,n}^T - \mathcal{F}_n \hat{\beta} \right)^T \Sigma_{m,n}^{-1} \left( y_{m,n}^T - \mathcal{F}_n \hat{\beta} \right) \right\},
\]

(1.3.36)

and the REML estimates are

\[
\left( \tilde{A}, \tilde{\theta} \right) = \arg\max \left\{ -\frac{1}{2} \log(|\Sigma_{m,n}|) - \frac{1}{2} \log(\mathcal{F}_n^T \Sigma_{m,n}^{-1} \mathcal{F}_n) - \frac{1}{2} \left( y_{m,n}^T - \mathcal{F}_n \tilde{\beta} \right)^T \Sigma_{m,n}^{-1} \left( y_{m,n}^T - \mathcal{F}_n \tilde{\beta} \right) \right\},
\]

(1.3.37)

where both maxima are over the assumed form for \( A \) and \( \hat{\beta} \) is given by (1.3.10). When numerically computing REML or maximum likelihood estimates of \( A \), we can express \( A \) as \( A = LL^T \), where \( L \) is a lower triangular matrix with positive diagonal elements. \( L \) has a 1-1 relationship with \( A \), and the simple constraints on the diagonals of \( L \) are much easier to deal with computationally than the positive-definiteness constraint on \( A \).

Notice that we have taken \( A \) in (1.3.28) to be the symmetric matrix square root of \( \Sigma_0 \). This is the approach advocated by Fricker et al. (2010). However, the geostatistics literature suggests modeling \( A \) in (1.3.28) as a lower triangular matrix rather than a symmetric one (see Gelfand et al. (2004), Banerjee and Johnson (2006) and Banerjee et al. (2008)). In this case, \( A \) is the Cholesky decomposition of \( \Sigma_0 = \text{Cov}(Y(x), Y(x)) \). The rationale is that a positive definite matrix has a 1−1 relationship with its Cholesky decomposition, so the Cholesky decomposition is just
as good as any interpretation of a matrix square root. Unfortunately, it appears that
this could be problematic, and produces a fundamentally different model than one
that assumes \( A \) is symmetric.

To see why a lower triangular assumption for \( A \) can be problematic, consider the
following theorem proved in Fricker et al. (2010).

**Theorem 3.** Suppose that (a) there exists \( 1 < j < k \) such that the \( j^{th} \) row of \( A \) has
non-zero entries to the left of the diagonal, and (b) the set of correlation functions
\( R_1(\cdot; \theta_1), \ldots, R_m(\cdot; \theta_m) \) for \( Z(x) \) have the property that that for some \( x, x' \in X \)
with \( x \neq x' \), and some \( u, v \in \{ 1, \ldots, j \} \), \( R_u(x - x'; \theta_u) \neq R_v(x - x'; \theta_v) \). Then, there
exist \( i < j \) such that

\[
\begin{align*}
\text{cov} (Y_i(x'), Y_j(x)|Y_i(x)) &= 0 \quad (1.3.38) \\
\text{cov} (Y_j(x'), Y_i(x)|Y_j(x)) &\neq 0 \quad (1.3.39)
\end{align*}
\]

*Proof.* See Fricker et al. (2010).

The implication of this results is that the use of a lower triangular \( A \) induces arti-
ficial asymmetry into the Gaussian process model. More specifically, \( Y_i(x') \) and \( Y_j(x) \)
are conditionally independent given \( Y_i(x) \), but, asymmetrically, \( Y_j(x') \) and \( Y_i(x) \) are
conditionally dependent given \( Y_j(x) \). So, if we have observed \( Y_i(x) \), then \( Y_j(x) \) gives
no further information regarding the value of \( Y_i(x') \), but if we have observed \( Y_j(x) \),
then \( Y_i(x) \) does give us some information regarding the value of \( Y_j(x') \). Such an
ordering of the codes is only appropriate if there is a natural hierarchy to the output,
as in Kennedy and O'Hagan (2000)). Such an assumption is not tenable in a general
multiple-output setting.
Table 1.1: ERMSPE’s for the WSNL function when we assume $A$ is lower triangular in the NLMC. Notice that predictive ability of the NLMC changes if we swap the order of $y_1(\cdot)$ and $y_2(\cdot)$.

<table>
<thead>
<tr>
<th>Function</th>
<th>Order</th>
<th>Model</th>
<th>Function</th>
<th>ERMSPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(y_1(\cdot), y_2(\cdot))$</td>
<td>NLMC</td>
<td>$y_1(\cdot)$</td>
<td>228.49</td>
<td></td>
</tr>
<tr>
<td>$(y_1(\cdot), y_2(\cdot))$</td>
<td>NLMC</td>
<td>$y_2(\cdot)$</td>
<td>17.80</td>
<td></td>
</tr>
<tr>
<td>$(y_2(\cdot), y_1(\cdot))$</td>
<td>NLMC</td>
<td>$y_1(\cdot)$</td>
<td>271.58</td>
<td></td>
</tr>
<tr>
<td>$(y_2(\cdot), y_1(\cdot))$</td>
<td>NLMC</td>
<td>$y_2(\cdot)$</td>
<td>22.11</td>
<td></td>
</tr>
<tr>
<td>$(y_1(\cdot), y_2(\cdot))$</td>
<td>Independence</td>
<td>$y_1(\cdot)$</td>
<td>228.51</td>
<td></td>
</tr>
<tr>
<td>$(y_1(\cdot), y_2(\cdot))$</td>
<td>Independence</td>
<td>$y_2(\cdot)$</td>
<td>22.11</td>
<td></td>
</tr>
<tr>
<td>$(y_2(\cdot), y_1(\cdot))$</td>
<td>Independence</td>
<td>$y_1(\cdot)$</td>
<td>228.51</td>
<td></td>
</tr>
<tr>
<td>$(y_2(\cdot), y_1(\cdot))$</td>
<td>Independence</td>
<td>$y_2(\cdot)$</td>
<td>22.11</td>
<td></td>
</tr>
</tbody>
</table>

For a specific example, suppose we use the different approaches ($A$ symmetric vs. $A$ lower triangular) to fit a Gaussian stochastic process model to the WSNL function described in Section 3.1.2. We will fit the independence model and NLMC (under both choices for $A$) using a training data set that consists of 15 evaluations of the WSNL function. The fifteen inputs form a maximin Latin hypercube design. We fit all models using both the ordering $(y_1(\cdot), y_2(\cdot))$, and, after swapping, the ordering $(y_2(\cdot), y_1(\cdot))$. All unknown covariance parameters will be estimated via REML. We will judge the quality of each model by calculating the estimated root mean square prediction error (ERMSPE) for each output, defined as

$$ERMSPE(\hat{y}_j(\cdot)) = \sqrt{\frac{\sum_{i=1}^{2500} (\hat{y}_j(x_i) - y_j(x_i))^2}{2500}},$$

where $\hat{y}_j(\cdot) = E[Y_j(\cdot)|\mathbf{y}^{m_\cdot n}]$ and $x_1, \ldots, x_{2500}$ is a dense grid over the input space.
Here are a few observations from our example. First, Table 1.1 shows how a lower triangular $A$ can lead to order-dependent predictions. Notice the difference in the predictions when we swap the ordering of the codes. Specifically, the swapped ordering results in much poorer prediction in terms of ERMSPE for both outputs. In contrast, notice in Table 1.2 that the predictions obtained using a symmetric $A$ matrix are unaffected by the ordering of $Y_1(\cdot)$ and $Y_2(\cdot)$. Second, the dependence model produces better predictions (in terms of the estimated root mean square prediction error) than the independence model in the symmetric $A$ case. Therefore, there are two key conclusions that can be made from this example. One, the predictive ability of a Gaussian process model is dependent on the ordering of the outputs if we assume $A$ is a lower triangular. Two, if we use a symmetric $A$ in an NLMC, it is possible to obtain better predictions than with an independence model, as we can see in Table
Table 1.2: ERMSPE’s for the WSNL function when we assume $A = LL^{\top}$ in the NLMC, where $L$ is lower triangular. Notice predictions are (practically) unchanged if we swap the order of $y_1(\cdot)$ and $y_2(\cdot)$ in the NLMC.

<table>
<thead>
<tr>
<th>Function Order</th>
<th>Model</th>
<th>Function</th>
<th>ERMSPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(y_1(\cdot), y_2(\cdot))$</td>
<td>NLMC</td>
<td>$y_1(\cdot)$</td>
<td>228.11</td>
</tr>
<tr>
<td>$(y_1(\cdot), y_2(\cdot))$</td>
<td>NLMC</td>
<td>$y_2(\cdot)$</td>
<td>17.19</td>
</tr>
<tr>
<td>$(y_2(\cdot), y_1(\cdot))$</td>
<td>NLMC</td>
<td>$y_1(\cdot)$</td>
<td>228.07</td>
</tr>
<tr>
<td>$(y_2(\cdot), y_1(\cdot))$</td>
<td>NLMC</td>
<td>$y_2(\cdot)$</td>
<td>17.19</td>
</tr>
<tr>
<td>$(y_1(\cdot), y_2(\cdot))$</td>
<td>Independence</td>
<td>$y_1(\cdot)$</td>
<td>228.51</td>
</tr>
<tr>
<td>$(y_1(\cdot), y_2(\cdot))$</td>
<td>Independence</td>
<td>$y_2(\cdot)$</td>
<td>22.11</td>
</tr>
<tr>
<td>$(y_2(\cdot), y_1(\cdot))$</td>
<td>Independence</td>
<td>$y_1(\cdot)$</td>
<td>228.51</td>
</tr>
<tr>
<td>$(y_2(\cdot), y_1(\cdot))$</td>
<td>Independence</td>
<td>$y_2(\cdot)$</td>
<td>22.11</td>
</tr>
</tbody>
</table>

1.2 and Figures 1.1-1.2. Notice that particularly in some of the corners of the input space, the squared prediction error of the NLMC is much smaller than the squared prediction error of the independence model.

**Other Dependence Models**

The nonseparable linear model of coregionalization (NLMC) and independence model are not the only covariance structures available. We will review several other ways one could model the dependence structure in multiple-output computer code.

The first alternative dependence model we will describe is the *nonseparable convolution emulator* (NCE) of Higdon (2002) and Fricker et al. (2010). This model assumes that each element of $Z(\mathbf{x})$ in (1.3.28) is constructed as

$$Z_i(\mathbf{x}) = \int_{\mathcal{X}} \kappa_i(\mathbf{u} - \mathbf{x}) W_i(\mathbf{u}) d\mathbf{u} \quad (1.3.40)$$
where $W^T(x) = [W_1(x), \ldots, W_m(x)]$ is a multivariate white noise process with a mean $\mathbf{0}_m$ and $\text{cov}[W_i(x), W_j(x')] = p_{i,j} \delta(x - x')$, where $p_{i,j}$ are elements of an $m \times m$ correlation matrix, and each $\kappa_i(\cdot)$ is a squared exponential smoothing kernel

$$
\kappa_i(x) = \sigma_i \left[ \left( \frac{4}{\pi^2} \right)^{d/2} \prod_{l=1}^{d} \Phi_{i,l}^{1/4} \exp \left\{ \sum_{l=1}^{d} \Phi_{i,l} (x_i - x'_l)^2 \right\} \right].
$$

(1.3.41)

We assume that each $\sigma_i > 0$ and each $\Phi_{i,j} > 0$. Letting $\Phi_i = \text{diag}(\Phi_{i,1}, \ldots, \Phi_{i,d})$, one can show that

$$
\text{cov} (Y_i(x), Y_j(x')) = \tilde{\Sigma}_{i,j} \rho_{i,j} \exp \left\{ -2 (x - x')^\top \Phi_i (\Phi_i + \Phi_j)^{-1} \Phi_j (x - x') \right\}.
$$

(1.3.42)

where $\tilde{\Sigma}_{i,j} = \sigma_i \sigma_j p_{i,j}$ and

$$
\rho_{i,j} = 2^{d/2} \prod_{l=1}^{d} \left( \Phi_{i,l} \Phi_{j,l} \right)^{1/4} \left( \Phi_{i,l} + \Phi_{j,l} \right)^{-1/2}.
$$

(1.3.43)

The unknown parameters that need to be estimated in this model are $\tilde{\Sigma}$, the $m \times m$ symmetric matrix whose $(i,j)$th element is $\tilde{\Sigma}_{i,j}$ and the $m \times m$ diagonal matrices $\Phi_1, \ldots, \Phi_m$.

How does the NCE compare to the NLMC? Both are equally complex, as they both contain $\frac{m(m+1)}{2} + dm$ unknown covariance parameters. However, each model’s parameters control different parts of the overall covariance relationship of the multivariate Gaussian process. In the NLMC, the parameters in $A$ directly control the between-output covariance relationship, while $\theta$ only indirectly controls the within-output (spatial) correlation. Conversely, in the NCE model, the parameters $\Phi_i$, $1 \leq i \leq m$, directly control the within-output correlation, but $\tilde{\Sigma}$ offers only limited control over the between-output covariance. Thus, in some sense, there is a trade-off between the two models. Empirically, how do they compare? Based on two case-studies, Fricker et al. (2010) concludes that the NCE model is inferior to the NLMC model in terms
of predictive ability of the $m$ outputs and in the predictive ability of functions of the $m$ outputs.

The next alternative model we review is the *spatially varying linear model of coregionalization* (SVLMC), which is discussed in Gelfand et al. (2004), Banerjee and Johnson (2006) and Banerjee et al. (2008). In this model, $A$ in (1.3.28) is replaced by $A(x)$, so that the between output covariance now depends on $x$, rather than remaining constant over the entire input space. To construct a spatially varying covariance, one assumes that $(A(x)A(x))^{-1}$ is a spatial Wishart process. Gelfand et al. (2004) describes how such process can be constructed as a product of two matrices whose elements are i.i.d. univariate Gaussian processes. While such an approach is much less restrictive than the NLMC, fitting such models considerably increases the computational demands.

The *separable linear model of coregionalization* (SLMC), described in Conti and O’Hagan (2010) and Fricker et al. (2010), is a computationally simpler alternative to the NLMC. The SLMC assumes that $Z(x)$ in (1.3.28) is an $m \times 1$ vector of not only independent, but *identically distributed* unit variance, zero mean Gaussian processes with common correlation function $R(x - x' ; \theta)$. Such an assumption makes the model mathematically and computationally more tractable. Under the i.i.d. assumption for the elements of $Z(x)$, the covariance matrix of the initial design is now has the simpler Kronecker product structure

$$\text{Cov}(Y^m, Y^m) = \Sigma_0 \otimes V$$

21
where $V$ is the $n \times n$ matrix with $(i,j)^{th}$ element $[V]_{i,j} = R(x_i - x_j; \theta)$. Moreover, if one assumes the improper prior

$$
\pi (\beta, \Sigma_0) \propto |\Sigma_0|^{-(m+1)/2},
$$

for $\beta$ and $\theta$, then one can analytically integrate $\Sigma_0$ out of the posterior distribution of $[Y(x) | Y^{m,n} = y^{m,n}]$, yielding a multivariate Student-$t$ posterior distribution that only depends on the correlation parameters $\theta$. Therefore, an implementation of the model would only require likelihood-based estimates of the $d$ elements of $\theta$.

This is in sharp contrast to the NLMC, which requires likelihood based estimates of $m(m+1)/2 + dm$ elements in $A, \theta_1, \ldots, \theta_m$. Unfortunately, such simplification makes the strong assumption that all $Y_1(\cdot), \ldots, Y_m(\cdot)$ share the same correlation parameters $\theta$.

Conti and O’Hagan (2010) used the SLMC to emulate a dynamic simulator. In this application, $y(\cdot) = (y_1(\cdot), \ldots, y_m(\cdot))$ represent a single complex system evolving over $m$ different time periods, and the assumption of the same correlation structure for each $Y_k(\cdot)$ seems reasonable. In other settings where each of the $m$ outputs could be a completely different complex system, such an assumption might not be tenable.

Fricker et al. (2010) contains case studies that illustrate the ineffectiveness of the SLMC as an emulator in a general multiple-output setting, as well as some deeper theoretical arguments against using the SLMC in general multiple-output settings.

### 1.4 Optimization of Single-Output Computer Experiments

Suppose we have an expensive single-objective black-box function (e.g., a computer experiment) $y(\cdot)$ with a $d$-dimensional input space $X$. Schonlau (1997) and Jones et al. (1998) introduced a popular sequential design strategy, referred to as the efficient global optimization (EGO) algorithm, for minimizing such a function based
on stochastic process models commonly employed in the statistics literature. The key elements of their procedure are a predictor of the function \( y(\cdot) \) at any given input, an estimate of the error in this predictor, and the expectation of a probabilistic improvement function based on the uncertainty of the predictor. Here, we will briefly review these concepts and present the EGO algorithm.

Suppose that we have evaluated \( y(\cdot) \) at \( D_n = (x_1, \ldots, x_n) \subset \mathcal{X} \). Denote \( y^n = (y(x_1), \ldots, y(x_n))^\top \) as the vector of outputs at \( D_n \). We model \( y(\cdot) \) as a Gaussian stochastic process with mean \( \beta \), variance \( \sigma^2 \), and correlation function

\[
R(x, x'; \rho, \theta) = \exp \left\{ \sum_{i=1}^{d} \theta_i |x_i - x'_i|^{\rho_i} \right\}. \tag{1.4.1}
\]

Let \( R \) be an \( n \times n \) matrix with \( R_{ij} = R(x_i, x_j; \rho, \theta) \). For a given input \( x \), let \( r \) be an \( n \times 1 \) vector with \( r_i = R(x, x_i; \rho, \theta) \).

If the covariance parameters \( \sigma^2, \theta, \rho \) are known, it can be shown (see Sacks et al. (1989)) that the best linear unbiased predictor (BLUP) of the function \( y(\cdot) \) at \( x \), conditional on the previous evaluations \( y^n \), is

\[
\hat{y}(x) = \hat{\beta} + r' R^{-1} (y^n - 1\hat{\beta}) \tag{1.4.2}
\]

where

\[
\hat{\beta} = \frac{1^\top R^{-1} y^n}{1^\top R^{-1} 1}. \tag{1.4.3}
\]

Also, the mean square prediction error (MSPE), which can be used to assess the accuracy of \( \hat{y}(x) \), can be shown to be

\[
s^2(x) = \sigma^2 \left[ 1 - r^\top R^{-1} r + \frac{(1 - 1^\top R^{-1} 1)^2}{1^\top R^{-1} 1} \right]. \tag{1.4.4}
\]
Unfortunately, in practice, $\sigma^2$, $\theta$ and $\rho$ are unknown and therefore are estimated via maximum likelihood. The estimator of $\sigma^2$, given $\theta$ and $\rho$, is

$$\hat{\sigma}^2 = \frac{(y^n - 1\hat{\beta})^\top R^{-1}(y^n - 1\hat{\beta})}{n}. \quad (1.4.5)$$

The estimators $\hat{\theta}$ and $\hat{\rho}$ of $\theta$ and $\rho$ can be found by numerically maximizing the profile likelihood function

$$\frac{1}{(2\pi)^{n/2}(\hat{\sigma}^2)^{n/2}|R|^{1/2}} \exp \left\{ -\frac{(y^n - 1\hat{\beta})^\top R^{-1}(y^n - 1\hat{\beta})}{2\hat{\sigma}^2} \right\}. \quad (1.4.6)$$

From here on, we will assume that these estimated covariance parameters have been substituted for their true but unknown values in (1.4.2)-(1.4.4).

The uncertainty in the true value of $y(\cdot)$ at $x$ is modeled as a normally distributed random variable $Y(x)$ with mean $\hat{y}(x)$ and variance $s^2(x)$. Define the improvement at $x$ as

$$I(x) = (y_{\text{min}}^n - Y(x)) 1_{[y_{\text{min}}^n - Y(x) > 0]}, \quad (1.4.7)$$

where $y_{\text{min}}^n$ is the smallest element in $y^n$. Notice that $I(x)$ is random variable, as it depends on $Y(x)$. The expected improvement is then defined as

$$E[I(x)] = \left[ (y_{\text{min}}^n - \hat{y}(x)) \Phi \left( \frac{y_{\text{min}}^n - \hat{y}(x)}{s(x)} \right) 
+ s(x) \phi \left( \frac{y_{\text{min}}^n - \hat{y}(x)}{s(x)} \right) \right] 1_{[s(x) > 0]} \quad (1.4.8)$$

Now, we can briefly outline the EGO algorithm.

1. Evaluate $y(\cdot)$ at an initial space-filling design $D_n = (x_1, \ldots, x_n) \subset \mathcal{X}$, such as a maximin Latin hypercube.

2. Obtain estimates of the stochastic process parameters $\beta$, $\sigma^2$, $\theta$ and $\rho$ based on the data $y^n$. These estimates can be used to calculate $\hat{y}(x)$ and $s^2(x)$ at any input $x \in \mathcal{X}$. 

24
3. Find $x^{n+1} = \arg \max E[I(x)]$. Notice that $E[I(x)]$ will be large if $s(x)$ is large (which corresponds to a large amount of uncertainty in the value of $y(x)$) or if the predicted value $\hat{y}(x)$ is much smaller than $y_{\min}^n$.

4. Evaluate $y(\cdot)$ at $x^{n+1}$. Repeat this process with $D_{n+1} = (x_1, \ldots, x_{n+1})$ and $y^{n+1} = (y(x_1), \ldots, y(x_n), y(x_{n+1}))^\top$ until a stopping criterion has been met.

Notice that the assumed distribution for $Y(x)$ is equivalent to the posterior distribution of $[Y(x)|Y^n = y^n]$ if a uniform prior is assumed for $\beta$ and all covariance parameters are known. Thus, the expected improvement function $E[I(x)]$ can be viewed as the expectation of $I(x)$ with respect to the posterior distribution of the Gaussian process modeling the computer code. From a Bayesian point of view, the third step of the EGO algorithm finds $x^{n+1}$ to maximize the posterior expected improvement $E[I(x)|Y^n = y^n]$.

Several generalizations of the EGO algorithm have been presented to handle a broader class of single-objective optimization problems. Schonlau et al. (1998) generalize the EGO algorithm to minimize the objective functions subject to bounds on several constraint functions, and also introduces the generalized improvement function

$$I^g(x) = (y_{\min}^n - Y(x))^g 1_{[y_{\min}^n - Y(x) > 0]},$$

where $g \geq 0$. Large values of $g$ lead to a more global search, and small values of $g$ produce a more local search. Williams et al. (2000) generalize the EGO algorithm to find the minimum of an integrated response function, which is of interest if some inputs are environmental variables that cannot be controlled in the physical process being
modeled by the computer code but have values governed by some probability distribution. Williams et al. (2010) further generalize the results of Williams et al. (2000) and Schonlau et al. (1998) by introducing methodology for minimizing one integrated response function subject to a constraint on another integrated response function. Huang et al. (2006) introduce a generalization of the EGO algorithm that allows one to optimize $y(\cdot)$ when the observations are corrupted by noise or measurement error.

1.5 Multiobjective Optimization

1.5.1 Basic Concepts and Terminology

Here, a brief overview of multiobjective optimization will be given. The terminology introduced will be used throughout the remainder of this thesis. Most of this material is drawn from Coello et al. (2006).

Let $\mathbf{y}(\cdot) = (y_1(\cdot), \ldots, y_m(\cdot))$ be an $m-$dimensional vector valued function ($m > 1$) from $\mathcal{X}$ to $\mathcal{Y}$, where the input space $\mathcal{X}$ is a $d$-dimensional rectangle in $\mathbb{R}^d$, and $\mathcal{Y} \subseteq \mathbb{R}^m$. (Usually, we will scale the input space to $[0,1]^d$.) The goal is to minimize $\mathbf{y}(\cdot)$ over the input space $\mathcal{X}$. Ideally, we would like to find a single $x$ that minimizes each $y_i(x)$, $1 \leq i \leq m$. In general, it is not possible to find a single solution to a multiobjective optimization problem. Instead, we will attempt to find a set of compromise or “trade-off” solutions.

The basic concepts required to describe a solution to a multiobjective optimization problem are as follows. For two inputs $x_1$ and $x_2$, we say $x_1$ weakly dominates $x_2$ ($x_1 \succeq x_2$) if $y_i(x_1) \leq y_i(x_2)$ for all $i = 1, \ldots, m$. If at least one inequality is strict, then $x_1$ is said to dominate $x_2$ ($x_1 \succ x_2$). We can also define dominance in a similar fashion for function outputs. For two elements $\mathbf{y}(x_1)$ and $\mathbf{y}(x_2)$ in the objective
space, we say $y(x_1)$ weakly dominates $y(x_2)$ ($y(x_1) \succeq y(x_2)$) if $y_i(x_1) \leq y_i(x_2)$ for all $i = 1, \ldots, m$. If at least one inequality is strict, then $y(x_1)$ is said to dominate $y(x_2)$ ($y(x_1) \succ y(x_2)$). An input vector $x \in \mathcal{X}$ is Pareto optimal if and only if there is no $x' \in \mathcal{X}$ such that $x \prec x'$. (Such $x$ are also referred to as nondominated inputs. The image $y(x)$ of a nondominated input is sometimes referred to as a non-dominated output.) The set of all Pareto optimal points in $\mathcal{X}$ is referred to as the Pareto set. Denote this set as $\mathcal{P}_X$. The corresponding image of $\mathcal{P}_X$ in the objective space is referred to as the Pareto front, and it is denoted $\mathcal{P}_Y$. In multiobjective optimization, the goal is to find (or, more realistically, approximate) the possibly uncountable sets $\mathcal{P}_Y$ and $\mathcal{P}_X$.

### 1.5.2 Pareto Set Approximation Quality Indicators

We will now introduce several metrics for comparing solutions to multiobjective optimization problems. These metrics rely on a the notion of an approximation set and a generalization of Pareto dominance for sets of points, rather than just single points, in the input or output space.

Let $A$ be a set of points in the objective space. $A$ is an approximation set if none of its elements dominate each other. In a 1–dimensional setting ($m = 1$), an approximation set would just be our best estimate of the true minimum of some function. In this situation, it is easy to compare two competing minima, because of the inherent ordering of $\mathbb{R}$. In a multiobjective problem, though, this is much more complicated, and we must rely on the notion of Pareto dominance to compare sets.

A set $A$ in the objective space weakly dominates the set $B$ in the objective space ($A \succeq B$) if every vector in $B$ is weakly dominated by some vector in $A$. The set $A$
dominates $B$ ($A \succ B$) if every vector in $B$ is dominated by a vector in $A$. These terms can be defined analogously for sets of points in the input space. Obviously, an approximation set $B$ that is weakly dominated or dominated by another approximation set $A$ would be considered an inferior approximation to the Pareto front.

Unfortunately, for any two approximation sets $A$ and $B$, neither one need dominate the other. To help determine how “good” an approximation set is, researchers studying multiobjective optimization have introduced several Pareto set approximation quality indicators. A detailed survey of this topic can be found in Zitzler et al. (2008).

Pareto set approximation quality indicators play two important roles in this thesis. First, we will use them as metrics for comparing various approaches to approximating the Pareto front of a computer experiment with multiple outputs. Second, many previously proposed approaches for the multiobjective optimization of computer experiments, as well as all of the new approaches we will propose in chapter 2, are based on these Pareto set approximation quality indicators. In the proceeding subsections, we will introduce four Pareto set approximation quality indicators that will be used extensively hereafter.

**Hypervolume Indicator**

This quality indicator was first introduced in Zitzler and Thiele (1998), and remains one of the most popular quality indicators in the multiobjective optimization literature. To calculate the hypervolume indicator for a set $A$ in the objective space, we first need to define a reference point $\mathbf{R}$, which is a point weakly dominated by all
vectors in \( \mathcal{Y} \). Define the hypervolume indicator of \( A \) as

\[
I_H(A, \mathbf{R}) = \int_{\mathbb{R}^m} 1\{ y \geq \mathbf{R}, \mathbf{A} \geq \{ y \} \} \, dy
\]

Basically, it is the Lebesgue measure of the set of points in the objective space that are dominated by \( A \) but also dominate some reference point \( \mathbf{R} \). Figure 1.3 illustrates \( I_H(A, \mathbf{R}) \) in the two-dimensional case.

![Diagram](image)

Figure 1.3: The circles form a 5-point set \( A \) in the objective space, the square is the reference point \( \mathbf{R} \), and the area of the gray shaded region is \( I_H(A, \mathbf{R}) \).

The hypervolume indicator’s main advantage is that it is strictly monotonic with respect to Pareto dominance. This means that if we have two approximation set \( A \) and \( B \) and \( A \succ B \), then \( I_H(A, \mathbf{R}) > I_H(B, \mathbf{R}) \). While there are many Pareto
set approximation indicators that are monotonic i.e., \( A \succeq B \), then \( I_H(A, R) \geq I_H(B, R) \), Zitzler et al. (2008) states that the hypervolume indicator and the weighted hypervolume indicator (which will be introduced shortly) are the only known Pareto set approximation quality indicators that are \textit{strictly} monotonic.

However, there are three disadvantages of the hypervolume indicator. One, it is dependent on the scaling of the various outputs. Two, it requires some additional problem knowledge, as it requires one to specify a suitable upper bound \( R \) on the output space. Three, it has a high computational cost. 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 objectives functions. Therefore, for even moderate sized objective spaces, one must usually resort estimating the hypervolume indicator via Monte Carlo methods.

\textbf{Weighted Hypervolume Indicator}

The next Pareto set approximation quality indicator can be viewed as a generalization of the hypervolume indicator. It was originally introduced in Zitzler et al. (2007) as a way of articulating user preference into a set quality indicator. To define this quality indicator, let \( w : \mathbb{R}^m \to \mathbb{R} \) be a nonnegative weight function, perhaps a probability density function, and let \( A \) be a set in the objective space. We define the weighted hypervolume indicator of \( A \) to be

\[
I_H^w(A) = \int_{\mathbb{R}^m} w(y) 1_{[A \succeq \{y\}]} dy. \tag{1.5.1}
\]

Auger et al. (2009) shows how different choices of \( w(\cdot) \) can express preference for the extremes of particular objectives or preference for particular regions of the objective space. For example, a multivariate normal pdf centered at \( \mu \) could be used
if the decision maker prefers the region around $\mu$ over other regions of the objective space. Sets in the objective space with many points near or dominating this preferred point would obtain larger weighted hypervolume measures than regions that are far away from the preferred point.

The main advantage of the weighted hypervolume indicator is that it maintains the strict monotonicity of the hypervolume indicator (see Zitzler et al. (2007)) and, depending on the choice of weight function, might no longer require a reference point $R$. (Of course, the choice of weight function requires additional problem knowledge as well.) Unfortunately, $I^w_{\mu}(A)$ is still expensive to compute exactly, and needs to be calculated via Monte Carlo methods in most cases. Additionally, it is still dependent on the scaling of the various outputs.

**Additive Binary-$\epsilon$ Indicator**

This Pareto set approximation quality indicator was introduced in Zitzler et al. (2003). Suppose $A$ and $B$ approximation sets. While the hypervolume based methods were defined for arbitrary sets in the objective space, notice that the additive binary-$\epsilon$ indicator requires that $A$ and $B$ are approximation sets. We define the additive binary-$\epsilon$ indicator to be

$$I_{\epsilon}^+ (A, B) = \inf_{\epsilon \in \mathbb{R}} \{ \forall y^2 \in B \exists y^1 \in A : y^1_i \leq \epsilon + y^2_i \forall i = 1, \ldots, m \}$$

This quantity is the smallest real number we must add to all elements of $B$ so that every element in $B$ is dominated by some element in $A$, i.e. $A \succ B$.

This quality indicator has the property that $I_{\epsilon}^+ (A, B) \leq 0$ if and only if $A \succeq B$. If $C$ is a fixed reference set, then $I_{\epsilon}^+ (A, C)$ is a monotonic, but not strictly monotonic, quality indicator, meaning that if $A \succeq B$, then $I_{\epsilon}^+ (A, C) \leq I_{\epsilon}^+ (B, C)$. One useful
choice for $C$, if it is known, is the true Pareto front $\mathcal{P}_Y$ (or, more realistically, a dense, discrete approximation of the true Pareto front). Then, we can think of $I_{\epsilon^+}(A, \mathcal{P}_Y)$ as a measure of the distance from the approximation set $A$ to the true Pareto front. In addition to its monotonicity, the additive binary-$\epsilon$ indicator is cheap to compute. Its major disadvantage, similar to the hypervolume and weighted hypervolume indicator, is that it is sensitive to the scaling of the output.

**Completeness Indicator**

The final quality indicator we will introduce is the *completeness indicator*, which is discussed in detail by Lotov (2005) and Zitzler et al. (2008). To define the completeness indicator, let

\[ A = \{ y(x_1), \ldots, y(x_n) \} \]

be a set of evaluations of the $m-$dimensional function $y(\cdot)$. The completeness indicator of $A$ is defined to be

\[ I_{CP}(A) = P(A \succeq \{ y(U) \}) \] (1.5.2)

where $U$ is a random variable distributed uniformly on the input space $X$. It can be viewed as a measure of the size of the region of $X$ dominated by the inputs associated with the elements of $A$. In this sense, it is similar to the hypervolume indicator, where the latter measures the size of the region of the output space dominated by $A$. Their are two major advantages of the completeness indicator, which are highlighted in in Zitzler et al. (2008). First, $I_{CP}(\cdot)$ is monotonic, but not strictly monotonic, with respect to Pareto dominance. Second, it is *scale invariant*. This means that if we substitute

\[ y'(\cdot) = \left( \frac{y_1(\cdot) - \mu_1}{\sigma_1}, \ldots, \frac{y_m(\cdot) - \mu_m}{\sigma_m} \right). \]
for $y(\cdot)$ and

$$A' = \{y'(x_1), \ldots, y'(x_n)\}$$

for $A$, where each $\mu_i \in \mathbb{R}$ and each $\sigma_i \in \mathbb{R}^+$, then

$$I_{CP}(A) = P(A \succeq \{y(U)\}) = P(A' \succeq \{y'(U)\}) = I_{CP}(A') \quad (1.5.3)$$

While all indicators discussed thus far respect Pareto dominance in some way, the completeness indicator is the only one that is not subject to change upon rescaling of the outputs. However, unless $y(\cdot)$ is analytically tractable, $I_{CP}(\cdot)$ cannot be calculated analytically and can only be approximated via Monte Carlo methods. Additionally, Zitzler et al. (2008) states that Monte Carlo estimates of $I_{CP}(A)$ are only effective for slowly-varying functions with input spaces with less than a dozen dimensions.

### 1.5.3 Classical Approaches to Multiobjective Optimization

Here, we will review two of the oldest approaches to approximating the Pareto set and front, as described in Coello et al. (2006).

The first approach is based on minimizing a linear combination of weights, and it is sometimes referred to as the *weighted-sum method*. In this approach, one simply minimizes a weighted sum of the objective functions

$$\sum_{i=1}^{m} w_i y_i(x), \quad w_i \geq 0 \text{ for } i = 1, \ldots, m \quad (1.5.4)$$

for a variety of weights $w_1, \ldots, w_m$. Under some regularity conditions, the optimal inputs for each combination of weights should correspond to a point in the Pareto set.
The second approach is called the $\epsilon$-constrained method. This method finds Pareto optimal inputs by solving the following optimization problem:

$$\min y_r(x)$$  \hspace{1cm} (1.5.5)

subject to: $y_k(x) \leq \epsilon_k$ for $k = 1, \ldots, m$ and $k \neq r$. \hspace{1cm} (1.5.6)

A set of Pareto optimal inputs can be generated by solving this problem for a variety of combinations of constraint levels $\epsilon_k$. In order to identify a proper set of $\epsilon_k$ values, one must normally carry out single-objective optimization on each output.

### 1.5.4 Multiobjective Evolutionary Algorithms (MOEAs)

A more modern approach for approximating the Pareto front and set through the multiobjective evolutionary algorithms (MOEAs). Because MOEAs encompass a large number of algorithms, an outline of one of the most popular MOEAs will be presented as opposed to a general outline. Specifically, the elitist non-dominated sorting genetic algorithm (NSGA-II) will be presented, which was first introduced in Deb et al. (2000). A more in-depth introduction to MOEAs can be found in Coello et al. (2006). Following the outline, some of the concepts will be described in more detail.

1. Choose a fixed number points $N$ in the input space in which to initially evaluate $y(\cdot)$. Label these initial inputs $P_0$.

2. Evaluate $y(\cdot)$ at $P_0$.

3. Compute the fitness of each individual in $P_0$, which is a measure of how “good” the current input is in terms of Pareto dominance. The fitness value of each individual depends on the other individuals in $P_0$, so the fitness is referred
to as the “shared” fitness. This fitness value, which will be described below, favors nondominated inputs over dominated ones, and encourages inputs whose outputs are not clustered together.

4. Using these initial inputs in $P_0$, $N$ new inputs should be generated using the evolutionary operations crossover and mutation. Individuals are chosen for crossover via binary tournament, a standard evolutionary selection. Crossover involves an exchange of information between two selected inputs, and mutation involves a random perturbation of newly generated inputs to encourage diverse inputs. Crossover, mutation, and selection are described in more detail below. Label this next generation of inputs $Q_0$.

5. Repeat the following procedure for $t \geq 0$ until a termination criterion has been met.

   (a) Evaluate $y(\cdot)$ at the inputs in $Q_t$.

   (b) Combine the parent and child populations to form $R_t = P_t \cup Q_t$, which is a population of size $2N$.

   (c) Create the population $P_{t+1}$ by choosing the $N$ elements in $R_t$ with the best fitness assignments.

   (d) $N$ new inputs are generated for $P_{t+1}$ using the evolutionary operations crossover and mutation. Individuals in $P_{t+1}$ are chosen for crossover via binary tournament selection.

Determining the fitness of each input in an MOEA can be done in a variety of ways. The NSGA-II algorithm uses nondominated sorting and a measure called crowding
distance to create a fitness ranking for all of the individual inputs. In this procedure, all inputs are first placed into different groups, where the group with rank 1 is the current Pareto set, rank 2 is the Pareto set if the rank 1 inputs are removed, and so on. Within each group, each input is assigned a crowding distance. For an input $\mathbf{x}$, the crowding distance is the average distance from $\mathbf{y}(\mathbf{x})$ to the two outputs on either side of this point along each of the objectives. So, all points in the group with rank 1 have a higher fitness ranking than the points in group 2 (and so on), and within the group with rank 1, the points with the larger crowding distance have higher fitness rank than those with a smaller crowding distance. Notice that upon termination of the algorithm, the rank 1 inputs constitute the approximation to the Pareto set.

During crossover, information is exchanged between two inputs in the current population $P_t$. While this can be done in a variety of ways, a classic example involves first converting two “parent” inputs, say $\mathbf{x}$ and $\mathbf{x}'$ to binary strings. In this case, suppose that

\[
\begin{align*}
\mathbf{x}^1 &= 00110100 \\
\mathbf{x}^2 &= 11100110
\end{align*}
\]

Then, a crossover point is chosen, and two “children” are created by swapping the bits beyond that point. For example, if the crossover point is between fourth and fifth bit, the children will be

\[
\begin{align*}
\mathbf{x}_c^1 &= 11100100 \\
\mathbf{x}_c^2 &= 00110110.
\end{align*}
\]
This is not the only way to perform crossover. Some approaches do not convert the inputs to binary form, such as the simulated binary crossover (see Agrawal et al. (1994)) used in NSGA-II.

How are inputs selected for crossover? One popular approach in the MOEA literature is a binary tournament. This is the approach used in NSGA-II. In this selection procedure, two candidates are randomly chosen from the current population $P_t$. The one that actually becomes the parent is the one with higher fitness ranking. Each parent for each crossover is selected in this manner.

The mutation operator randomly changes a child created via crossover. In the binary encoding above, this could be accomplished by randomly switching a bit with a given probability, so it goes from 0 to 1 or 1 to 0. For example, if each child is represented by an 8-bit string, then, for every child, we could generate eight uniform random numbers $U_1, \ldots, U_8$. If the mutation probability was 0.1, then would change the encoding of the $i^{th}$ bit of the current child if $U_i$ was greater than 0.9. We should note that mutation in NSGA-II is different than this classic approach that involves binary encoding, as NSGA-II keeps all input variables real-valued.

1.5.5 The Expected Improvement Approach

When function evaluations are expensive, classical approaches and MOEAs may not be feasible to solve multiobjective optimization problems. Both the weighted-sum and $\epsilon$-constrained method require that one solve multiple single-objective optimization problems, which would not be analytically tractable in many real-world settings, and therefore would require some sort of iterative method that requires many function
evaluations. MOEAs not only require a relatively large number of function evaluations, but also require fitness evaluations that, according to Emmerich et al. (2006), can also be computationally intensive to compute. One solution to the problem is to generalize the notion of “expected improvement” from the EGO algorithm described in Section 1.4. In this case, $y_{\text{min}}^n$, the current minimum among $n$ function evaluations, is replaced by the current Pareto front, denoted $\mathcal{P}_n^Y$, which is the set of nondominated outputs among the set of $n$ function evaluations of the $m$-dimensional function $y(\cdot)$. Also, one must define some sort of improvement function to quantify how much an output $y(x)$ improves upon $\mathcal{P}_n^Y$. Such generalization have been proposed by Emmerich et al. (2006), Keane (2006), and Bautista (2009). In Chapter 2, we will set up a general framework for the expected improvement approach to multiobjective optimization, overview the approaches from the literature, propose new methodology, and compare the various approaches based on theoretical considerations. In Chapter 3, we will compare these various generalization of the EGO algorithm based on a set test problems from the MOEA literature and real-world applications.

### 1.5.6 Other Approaches to Optimizing Multiple Output Computer Experiments

We will now describe other proposed approaches to optimizing multiple output computer code that exploit cheap-to-compute surrogates, such as Gaussian process models. Some of these approaches fall outside of the expected improvement framework, while others do not explicitly attempt to approximate the Pareto front and set.

Wilson et al. (2001) presents a simple approach to approximating the Pareto front when practical constraints do not allow for classical approaches or MOEAs. These
authors fit independent Gaussian process models to each output and then evaluate the BLUP $\hat{y}_1(x), \ldots, \hat{y}_m(x)$ of each output over a dense set of inputs. The nondominated predictions over this dense set of inputs are the estimated Pareto front, and the inputs corresponding to these outputs comprise the estimated Pareto set. This was the approach used to solve the piezoelectric bimorph actuator design problem described in Section 1.2.1.

Voutchkov and Keane (2006) introduces a sequential algorithm the only relies on the point predictions of the various outputs, and does not take into account the uncertainty of these predictions. In this approach, a surrogate (such as a Gaussian process model, radial basis function model, or polynomial approximation) is created for each objective function based on an initial set of training data. A multiobjective evolutionary algorithm is run on this collection of surrogates. Then, the objective functions are evaluated at the estimated Pareto set determined by these MOEA runs on the surrogate. The process is then repeated until a stopping criterion is attained. A weakness of this approach is that it can get “stuck” in local Pareto optimal sets and completely miss the global Pareto optimal set.

Knowles (2006) generalizes the EGO algorithm in a slightly different way in the so-called ParEGO algorithm. In this approach, a scalar utility function of the outputs at an initial design is created, where the weight of each output on this utility function is randomly chosen. At each iteration of the algorithm, a Gaussian process model is fit to this randomly weighted scalar function of the outputs. Then, the expected improvement of this scalar utility function is maximized, and the maximizer is chosen
as the next input. More specifically, the scalar utility function is defined to be

$$z_\lambda(x) = \max_{k=1,...,m} [\lambda_k y_k(x)] + 0.05 \sum_{k=1}^{m} \lambda_k y_k(x),$$  \hspace{1cm} (1.5.7)

and the vector of weights $\lambda = (\lambda_1, \ldots, \lambda_m)$ is chosen randomly from the set

$$\Lambda_s = \left\{ \lambda = (\lambda_1, \ldots, \lambda_m) \mid \sum_{k=1}^{m} \lambda_k = 1 \text{ and } \lambda_k \in \left\{ 0, \frac{1}{s}, \frac{2}{s}, \ldots, 1 \right\} \right\},$$  \hspace{1cm} (1.5.8)

where $s$ is a positive integer. Letting

$$z_\lambda^{\max} = \max\{z_\lambda(x_1), \ldots, z_\lambda(x_n)\}$$  \hspace{1cm} (1.5.9)

an interpolating Gaussian process model $Z_\lambda(\cdot)$ is assumed for $z_\lambda(\cdot)$. The next input is chosen in each iteration by maximizing the expected value of

$$I(x) = (Z_\lambda(x) - z_\lambda^{\max})^g 1[Z_\lambda(x) - z_\lambda^{\max} > 0]$$  \hspace{1cm} (1.5.10)

with respect to the distribution of $Z_\lambda(\cdot)$ conditional on

$$Z_\lambda(x_1) = z_\lambda(x_1), \ldots, Z_\lambda(x_n) = z_\lambda(x_n).$$

Notice that each iteration of the algorithm requires different randomly chosen weights for the various outputs. Because of its stochastic nature, the ParEGO algorithm is, in some sense, fundamentally different from the expected improvement approach to multiobjective optimization as it will be presented in Chapter 2.

Henkenjohann and Kunert (2007) proposes a multivariate generalization of the EGO algorithm with the goal of sequentially adding inputs to produce outputs near a particular “desirable” region of the objective space. This approach is a fundamentally different approach to multiobjective optimization, as it uses expert advice to focus on particular regions of the output space, rather than attempting to approximate the
entire set of Pareto optimal inputs and outputs. To obtain objective space vectors in these desirable regions, each output is assigned a desirability function \( d_i(y_i(x)) \), which is a parameterized function that expresses the quality of the particular output. For example, if the goal is minimization of each output, one choice of a desirability index is

\[
d_i(y_i(x)) = \exp \{- \exp \{- a_i + b_i y_i(x)\}\},
\]

where \( a_i \) and \( b_i \) are user-defined parameters that would express the importance of minimizing the \( i^{th} \) objective. A desirability index for a set of outputs at one particular input is created by taking the geometric mean of the desirability function values for the outputs, i.e.,

\[
DI(y(x)) = \prod_{i=1}^{m} d_i(y_i(x))^{1/m}.
\]

So, if there is an initial design \( x_1, \ldots, x_n \), one can calculate a desirability index \( DI(y(x_1)), \ldots, DI(y(x_n)) \) for each output at the initial design. The current most desirable input would be

\[
DI_{\max} = \max \{DI(y(x_1)), \ldots, DI(y(x_n))\}.
\]

Also, let

\[
F_{\max} = \{f : DI(f) = DI_{\max}\}.
\]

This is the subset of vectors in the desirability function space with a desirability index equal to \( DI_{\max} \). Additionally, let \( D(y(x)) = (d_1(y(x)), \ldots, d_m(y_m(x))) \) Then, the improvement function is defined as

\[
I_{DI}(x) = \begin{cases} 
\min_{f \in F_{\max}} \|f - D(y(x))\|^g & \text{if } DI(y(x)) \geq DI_{\max} \\
0 & \text{otherwise.}
\end{cases}
\]
for some \( g \geq 0 \). Therefore, inputs can be added sequentially by maximizing the expected value of \( I_{DI}(x) \) with respect to the distribution of \( Y(x) \) conditional on the initial function evaluations.

### 1.6 Sample Average Approximations

Throughout this thesis, when using the method 1 interpretation of \( QI(x) \), we will need to solve optimization problems of the form

\[
\max_{x \in \mathcal{X}} E \left[ I^*(Y(x)) | Y^{m,n} = y^{m,n} \right],
\]

where \( I^*(\cdot) \) is a multivariate generalization of the improvement function employed in Schonlau (1997) and Jones et al. (1998). In some situations, the expectation in (1.6.1) is analytically tractable, and we can optimize the expectation in a straightforward manner. However, in most instances, \( E[I^*(Y(x)) | Y^{m,n} = y^{m,n}] \) is intractable, and this thesis uses a sample average approximation (see Shapiro (2003)) to optimize this function.

The sample average approximation is a tool for solving stochastic programming problems of the form

\[
\max_{x \in \mathcal{X}} E[F(x, \xi)],
\]

where \( F(x, \xi) \) is function of \( x \in \mathcal{X} \subseteq \mathbb{R}^d \) and the \( m \)--dimensional random vector \( \xi \), and the expectation is taken with respect to the distribution of \( \xi \). The sample average approximation method generates a random sample \( \xi_1, \ldots, \xi_{N_{mc}} \) from the distribution of \( \xi \), and approximates the true optimization problem (1.6.2) by the optimization problem

\[
\max_{x \in \mathcal{X}} \left\{ \hat{f}_{N_{mc}}(x) = \frac{1}{N_{mc}} \sum_{i=1}^{N_{mc}} F(x, \xi_i) \right\}.
\]
The function \( \hat{f}_{Nmc}(\bm{x}) \) is referred to as the \textit{sample average function}. Notice that the sample average function is deterministic for a given realization of \( \xi_1, \ldots, \xi_{Nmc} \), and an approximate solution to (1.6.2) can be found by solving (1.6.3) using an appropriate deterministic optimization algorithm. This is the main advantage of the sample average approximation; after generating one random sample, we are able to employ any number of efficient, well-studied, and reliable optimization routines for solving deterministic optimization problems.

Notice that the maximizer \( \hat{\bm{x}}_{Nmc} \) of (1.6.3) and the associated maximum \( \hat{f}(\hat{\bm{x}}_{Nmc}) \) are statistical estimators of the true maximizer and true maximum. Under mild assumptions, Shapiro (2003) shows that these estimators are consistent as \( N_{mc} \to \infty \).

To construct a sample average approximation to solve (1.6.1), we can express \( I^*(\bm{Y}(\bm{x})) \) as

\[
I^*(\bm{Y}(\bm{x})) = I^*(C(\bm{x})\bm{Z} + \hat{\bm{y}}(\bm{x})),
\]

where \( C(\bm{x}) \) is the Cholesky decomposition of \( \bm{S}(\bm{x}) \) and \( \bm{Z} \sim N(\bm{0}_m, \bm{I}_m) \), so that \( I^*(\cdot) \) and \( \bm{Z} \) play the role of \( F(\cdot) \) and \( \bm{\xi} \), respectively, in (1.6.2).

### 1.7 Sensitivity Analysis and Sensitivity Indices

The goal of sensitivity analysis is to determine how variable the output of a computer experiment is to changes in the input. In this section, we will describe \textit{sensitivity indices}, a popular tool for performing sensitivity analysis, which are based on ANOVA-type function decompositions. Also, unlike previous sections, we now assume that our computer code output \( y(\bm{x}) = y(x_1, \ldots, x_d) \) is \textit{one-dimensional}. Assume the \( d \)-dimensional cube, \( \mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_d = [l_1, u_1] \times \ldots \times [l_d, u_d] \), is the input space.
The methods presented in this section require that we assume a joint probability density \( g(\mathbf{x}) = \prod_{k=1}^{d} g_k(x_k) \) on the inputs \( \mathbf{X} \). In the following, a general marginal weight function \( g_k(x_k) \) on \( \mathcal{X}_k = [l_k, u_k] \), \( k = 1, \ldots, d \), is used.

### 1.7.1 Sobol’ Decomposition

Let

\[
y_0 = E_g [y(\mathbf{X})] = \int_{\mathcal{X}} y(\mathbf{x}) g(\mathbf{x}) \, d\mathbf{x}
\]
denote the overall mean of \( y(\mathbf{x}) \). Sobol’ (1993) showed that there is a unique decomposition of \( y(\mathbf{x}) = y_0 + \sum_{i=1}^{d} y_i(x_i) + \sum_{1 \leq i < j \leq d} y_{ij}(x_i, x_j) + \ldots + y_{1,2,\ldots,d}(x_1, \ldots, x_d) \) (1.7.1) that satisfies

\[
E_g [y_{i_1,\ldots,i_s}(X_{i_1}, \ldots, X_{i_s})] = \int_{\mathcal{X}_{i_k}} y_{i_1,\ldots,i_s}(x_{i_1}, \ldots, x_{i_s}) g_k(x_{i_k}) \, dx_{i_k} = 0 \quad (1.7.2)
\]

for any \( 1 \leq i_k \leq s \), and that has orthogonal components; that is, for any \( (i_1, \ldots, i_s) \neq (j_1, \ldots, j_t) \),

\[
E_g [y_{i_1,\ldots,i_s}(X_{i_1}, \ldots, X_{i_s})y_{j_1,\ldots,j_t}(X_{j_1}, \ldots, X_{j_t})] = \int_{\mathcal{X}} y_{i_1,\ldots,i_s}(x_{i_1}, \ldots, x_{i_s})y_{j_1,\ldots,j_t}(x_{j_1}, \ldots, x_{j_t}) g(\mathbf{x}) \, d\mathbf{x} = 0. \quad (1.7.3)
\]

The component terms in (1.7.1) are obtained by defining

\[
y_i(x_i) = E_g [y(\mathbf{X}) \mid X_i = x_i] - y_0 = \int_{\mathcal{X}_{-i}} y(\mathbf{x}) \prod_{k \neq i} g_k(x_k) \, d\mathbf{x}_{-i} - y_0
\]
to be the “main effect” function of input \( x_i \), for \( 1 \leq i \leq d \), and

\[
y_{ij}(x_i, x_j) = E_g [y(\mathbf{X}) \mid X_i = x_i, X_j = x_j] - y_i(x_i) - y_j(x_j) - y_0
\]
\[
= \int_{\mathcal{X}_{-\{ij\}}} y(\mathbf{x}) \prod_{k \neq i,j} g_k(x_k) \, d\mathbf{x}_{-\{ij\}} - y_i(x_i) - y_j(x_j) - y_0
\]
to be the interaction effect function of inputs $x_i$ and $x_j$, where $1 \leq i < j \leq d$. Here $dx_{-i}$ denotes integration of all inputs except $x_i$ over the input region $X_{-i} = \prod_{k \neq i} X_k$ and, analogously, $dx_{-(ij)}$ denotes integration of all inputs except $x_i$ and $x_j$ over the input region $X_{-(ij)} = \prod_{k \neq i,j} X_k$. Higher-order interaction terms are defined in a recursive manner.

The total variance of $y(x)$ (with respect to $g(\cdot)$) is defined to be

$$V = \text{Var}_g[y(X)] = \int_X y^2(x) g(x) dx - y_0^2.$$  

Using (1.7.1)-(1.7.3), the variance of $y(x)$ can be partitioned as

$$V = \sum_{i=1}^d V_i + \sum_{1 \leq i < j \leq d} V_{ij} + \ldots + V_{1,2,\ldots,d} \quad (1.7.4)$$

where

$$V_i = \text{Var}_g(E_g[y(X) \mid X_i = x_i]) = \int_{X_i} y_i^2(x_i) g_i(x_i) dx_i,$$

$$V_{ij} = \text{Var}_g(E_g[y(X) \mid X_i = x_i, X_j = x_j]) = \int_{X_i \times X_j} y_{ij}^2(x_i, x_j) g_i(x_i) g_j(x_j) dx_i dx_j,$$

for $1 \leq i < j \leq d$. In general, the variance $V_{k_1,\ldots,k_s}$ is defined as

$$V_{k_1,\ldots,k_s} = \text{Var}_g[y_{k_1,\ldots,k_s}(X_{k_1},\ldots,X_{k_s})]$$

$$= \text{Var}_g(E_g[y(X) \mid X_{k_1} = x_{k_1}, \ldots, X_{k_s} = x_{k_s}])$$

$$= \int_{X_{k_1} \times \ldots \times X_{k_s}} y_{k_1,\ldots,k_s}^2(x_{k_1},\ldots,x_{k_s}) \prod_{i=1}^s g_k(x_{k_i}) dx_{k_1} \ldots dx_{k_s}$$

because each $y_{k_1,\ldots,k_s}$ has zero mean by construction for any subset $(k_1,\ldots,k_s) \in \{1,\ldots,d\}, 1 \leq s \leq d$.

The sensitivity of $y(\cdot)$ with respect to $(k_1,\ldots,k_s)$ is defined by

$$S_{k_1,\ldots,k_s} = \frac{V_{k_1,\ldots,k_s}}{V}. \quad (1.7.5)$$
The value $S_{k_1,...,k_s}$ measures the proportion of total variance explained by $(x_{k_1}, \ldots, x_{k_s})$ interactions. By (1.7.4),

$$
\sum_{i=1}^{d} S_i + \sum_{1 \neq i < j \leq d} S_{ij} + \ldots + S_{1,2,\ldots,d} = 1. 
$$

(1.7.6)

The quantities $S_1, \ldots, S_d$ are called the main effect sensitivity indices and $S_{ij}$ is a two-factor sensitivity index. The total effect sensitivity index of the input $x_i$ is defined to be the sum of all sensitivity indices involving the input $x_i$,

$$
T_i = S_i + \sum_{j \neq i} S_{ij} + \ldots + S_{1,2,\ldots,d}. 
$$

(1.7.7)

For example, if there are $d = 3$ inputs, then $T_1 = S_1 + S_{12} + S_{13} + S_{123}$. Notice that, by construction, $S_i \leq T_i$ for all $i \in \{1, \ldots, d\}$. If interactions involving $x_i$ account for a large proportion of the variance $V$, then the difference between $T_i$ and $S_i$ will be large.

### 1.7.2 Joint Effect Functions

When discussing main effect and higher-order effect terms, the following notation will be used. If the effect of a subset of input variables $S = \{k_1, \ldots, k_s\} \subset \{1, 2, \ldots, d\}$ is of interest, let $x_S$ denote the vector of inputs $(x_{k_1}, \ldots, x_{k_s})$. The vector of the remaining inputs will be denoted by $x_{-S}$. By rearranging the order of the input variables, one may write $x = (x_S, x_{-S})$. Similarly let $X_S$ denote the input region of the inputs $x_S$ and let $X_{-S}$ denote the input region of the remaining inputs. Then $X = X_S \times X_{-S}$.

Now a set of functions closely related to the corrected effect function $y_S(\cdot)$ are introduced. For any non-empty $S = \{k_1, \ldots, k_s\} \subset \{1, 2, \ldots, d\}$, define the joint
effect of the inputs \( x_S \) to be

\[
j_S(x_S) = E_g[y(X)|X_S = x_S] = \int_{x_S} y(x_S, x_{-S})g(x_{-S})dx_{-S}.
\] (1.7.8)

The joint effects \( j_S(x_S) \) need not be centered nor need \( j_{S_1}(x_{S_1}) \) and \( j_{S_2}(x_{S_2}) \) be orthogonal for \( S_1 \neq S_2 \). The joint effect and Sobol’ component are closely related. For example, the main effect of input \( x_i \) is

\[
y_i(x_i) = j_i(x_i) - y_0, \quad 1 \leq i \leq d \quad (1.7.9)
\]

and the interaction effect of inputs \( x_i \) and \( x_j \) satisfies

\[
y_{ij}(x_i, x_j) = j_{ij}(x_i, x_j) - j_i(x_i) - j_j(x_j) - y_0. \quad (1.7.10)
\]

Equation (1.7.10) shows that the joint effect \( j_{ij}(x_i, x_j) \) includes the interaction effects as well as the main effects of \( x_i \) and \( x_j \).

For arbitrary \( S \), let

\[
V^j_S = Var_g[j_S(X_S)] = Var_g[E_g[y(X)|X_S]] \quad (1.7.11)
\]

be the variance of the joint effect. The quantity \( V^j_S \) has the interpretation that it measures the expected reduction in uncertainty in \( y(X) \) due to observing \( x_S \), because

\[
V^j_S = Var_g[y(X)] - E_g[Var_g[y(X)|X_S]].
\]

Consider two special cases of the joint effect function. The variance of the effect function \( j_i(x_i) \) of the individual input \( x_i \) is

\[
V^j_i = Var_g[y_i(X_i) + y_0] = V_i \quad (1.7.12)
\]

from (1.7.9), so \( V^j_i \) is the same as \( V_i \) in (1.7.4). The main effect sensitivity index of input \( x_i, S_i, \) can be computed from
Using (1.7.10), (1.7.11), and (1.7.3), the variance of the joint effect of inputs \( x_i \) and \( x_j \) is

\[
V_{ij}^j = \text{Var}_g[y_i(X_i) + y_j(X_j) + y_{ij}(X_i, X_j) + y_0] = V_i + V_j + V_{ij}
\]

(1.7.14)
because the components are orthogonal. Equation (1.7.14) contains both the variance of the main effects and the variance of the interaction effect of inputs \( x_i \) and \( x_j \). Thus \( V_S^j \neq V_S \) when \( S \) contains more than one input.

Let \( X_{-i} \) denote the vector of all components of \( X \) except \( X_i \). Then

\[
V_{-i}^j = \text{Var}_g[j_{-i}(X_{-i})]
\]

\[
= \text{Var}_g[y_{1,2,...,i-1,i+1,...,d}(X_{-i}) + \ldots + y_1(X_1) + y_2(X_2) + \ldots + y_{i-1}(X_{i-1}) + y_{i+1}(X_{i+1}) + \ldots + y_d(X_d) + y_0]
\]

\[
= \text{Var}_g \left[ \sum_{S: i \notin S} y_S(X_S) \right]
\]

\[
= V_{1,2,...,i-1,i+1,...,d} + \ldots + V_1 + V_2 + \ldots + V_{i-1} + V_{i+1} + \ldots + V_d
\]

(1.7.15)
is the sum of all \( V_S \) components \textit{not involving} the subscript \( i \) in the variance decomposition (1.7.4). Thus \( V - V_{-i}^j \) is the sum of all \( V_S \) components that \textit{do involve} the input \( x_i \). The quantity \( V - V_{-i}^j \) can be used to compute the total effect sensitivity index (1.7.7) via the equation

\[
T_i = \frac{V - V_{-i}^j}{V}.
\]

(1.7.16)

Therefore, to estimate only the \textit{main effect} and \textit{total effect} sensitivity indices \( S_i \) and \( T_i \) in a computer experiment, one only needs to estimate joint effect variances. Estimates of the variances of the corrected effects in the Sobol’ decomposition are not
necessary. In Chapter 4, we will describe in detail how we can estimate sensitivity indices in the Gaussian process framework.

In addition to their usefulness in the calculation of main effect and total effect sensitivity indices, joint effect functions can also be useful tools for performing sensitivity analysis via graphical methods. A main effect plot displays displays \((x_i, j_i(x_i))\) over the range of the \(i^{th}\) variable for \(1 \leq i \leq d\), and an interaction plot displays \((x_i, x_j)\) versus \(j_{i,j}(x_i, x_j)\) over the range of the \(i^{th}\) and \(j^{th}\) variable for \(1 \leq i < j \leq d\). A large amount of variability in a main effect or interaction plot indicates a large amount of sensitivity to that particular variable (or combination of variables).
CHAPTER 2

THE EXPECTED IMPROVEMENT APPROACH FOR
MULTIOBJECTIVE OPTIMIZATION OF COMPUTER
EXPERIMENTS

In this chapter, a general framework will be presented for expected improvement algorithms for multiobjective optimization. The key components of this approach are the choice of improvement function, which quantifies how much $y(x)$ improves upon the current Pareto front $\mathcal{P}_y^n$, and the method by which one defines a so-called “expected improvement” function. Several proposed improvement functions from the literature will be reviewed, and three new improvement functions will be introduced. We will also discuss how one can implement these improvement functions using two different generalizations of the one-dimensional expected improvement function. Finally, we will introduce six desirable properties that an improvement function should have, and determine how each of the various improvement functions fare in regards to these desirable properties.

2.1 The Expected Improvement Algorithm

Multiobjective optimization is not easy. Like single-objective optimization, most real-world problems cannot be solved analytically, and we must rely on a numerical
method to obtain an approximate solution to a multiobjective optimization problem. Unlike single-objective optimization, a numerical solution, except for trivial or pathological cases, involves multiple points in the input and output space. Ideally, a solution involves a finite set of points that are evenly spread across the true Pareto front and Pareto set. A challenge in the computer experiments setting is that most popular methods for approximating the Pareto front, such as weighted sum approaches, $\epsilon$-constrained approaches, and multiobjective evolutionary algorithms (MOEAs) can require a large number of function evaluations. To deal with the expensive, black-box nature of a computer experiment, several generalizations of EGO have been proposed in the literature. Here, we will set-up a general framework for expected improvement algorithms for multiobjective optimization that can be used in expensive-to-compute settings.

2.1.1 Outline of the Algorithm

Assume that the $m$-dimensional function $\mathbf{y}(\cdot)$ a realization of an $m$-variate Gaussian process model $\mathbf{Y}(\cdot)$ as described in Section 1.3. All approaches we will review from the literature as well as the newly proposed approaches have the following general structure:

1. Evaluate $\mathbf{y}(\cdot)$ at an initial space-filling design $\mathcal{D}_n = (x_1, \ldots, x_n) \subset \mathcal{X}$. Let
   $$\mathbf{y}^{m,n} = (\mathbf{y}^\top(x_1), \ldots, \mathbf{y}^\top(x_n))^\top.$$

2. Estimate the unknown covariance parameters using maximum likelihood or REML based on the prior distribution associated with $\mathbf{y}^{m,n} = (\mathbf{y}^\top(x_1), \ldots, \mathbf{y}^\top(x_n))^\top$. 

51
3. Calculate the current Pareto set \( P_n^X \) and current Pareto front \( P_n^Y \). These are the set of nondominated inputs among all inputs in \( D_n \) and all nondominated outputs among all functions evaluations in \( y^{m,n} \), respectively.

4. Find \( x^{n+1} \) by maximizing a given quality improvement measure \( QI(x) \). The function \( QI(\cdot) \) will depend on the chosen improvement function \( I^*(y(x)) \), i.e., the multivariate generalization of \( I(x) \) from Section 1.4. Below, we will present two ways that \( QI(x) \) has been defined in the literature, as well as several proposed improvement functions. Note that \( QI(x) \) is commonly referred to as the “expected improvement” at \( x \), but, as we will see below, it need not be defined as the expectation of the chosen improvement function.

5. Evaluate \( y(x^{n+1}) \). If our budget has been exhausted or a stopping criterion has been met, terminate the algorithm. Otherwise, repeat steps 2 - 5 with \( y^{n+1,m} = (y^\top(x_1), \ldots, y^\top(x_n), y^\top(x_{n+1}))^\top \).

We will now present two methods for defining \( QI(x) \) in terms of a given improvement function \( I^*(y(x)) \):

- **Method 1**: Define the quality improvement associated with \( I^*(\cdot) \) to be

\[
QI(x) = E[I^*(Y(x))|Y^{m,n} = y^{m,n}].
\]  

(2.1.1)

\( QI(\cdot) \) is an expectation of the posterior predictive distribution, and thus it is natural to refer to this improvement measure as an expected improvement function.

- **Method 2**: Define the quality improvement associated with \( I^*(\cdot) \) to be

\[
QI(x) = P(Y(x) \in \mathcal{R}_n|Y^{m,n} = y^{m,n}) I^*(\bar{Y}(x))
\]  

(2.1.2)
where
\[ R_n = \{ y : y \not\preceq z, \forall z \in P^n \} \] \hspace{1cm} (2.1.3)
is the region of the objective space that is nondominated by the current Pareto front, \( Y(x) \) is the mean of the distribution of \( [Y(x_0)|Y^{m,n} = y^{m,n}, Y(x_0) \in R_n] \), which can be expressed as
\[
Y(x) = \frac{E\left[ Y(x) | Y(x) \in R_n, Y^{m,n} = y^{m,n} \right]}{P(Y(x) \in R_n | Y^{m,n} = y^{m,n})}. \hspace{1cm} (2.1.4)
\]
Notice that \( Y(x) \) is an \( m \)-dimensional vector.

Method 1 is the more natural multivariate extension of the single-objective expected improvement function. In this case, we define an improvement function, and compute its expectation conditional on the function evaluations we have already observed. In practice, we can attempt to work out an analytic expression for (2.1.1). If this is not possible, we can turn to Monte Carlo methods by simulating a large number of draws from the distribution of \( [Y(x_0)|Y^{m,n} = y^{m,n}, Y(x_0) \in R_n] \), evaluating the improvement function at each draw, and averaging these values. The strong law of large numbers will guarantee convergence to (2.1.1) at any given \( x \).

Method 2 is less natural. First and foremost, (2.1.2) is a function of expectations, but generally is not an expectation. Therefore, referring to it as an expected improvement function can be misleading. However, it plays the same role as the single-objective expected improvement function employed in Section 1.4, so we will continue to refer to it as an “expected improvement” function.

53
For output with only one objective, (2.1.2) is equivalent to the single-objective expected improvement function. When \( m = 1 \),

\[
E \left[ I(y(x)) | Y^{n,1} = y^{n,1} \right]
\]

\[
= E \left[ (y^{\text{min}} - Y(x)) 1_{Y(x) - y^{\text{min}} < 0} | Y^{n,1} = y^{n,1} \right]
\]

\[
= y^{\text{min}} E \left[ 1_{Y(x) - y^{\text{min}} < 0} | Y^{n,1} = y^{n,1} \right]
\]

\[
- E \left[ Y(x) 1_{Y(x) - y^{\text{min}} < 0} | Y^{n,1} = y^{n,1} \right]
\]

\[
= E \left[ 1_{Y(x) - y^{\text{min}} < 0} | Y^{n,1} = y^{n,1} \right]
\]

\[
\times \left( y^{\text{min}} - E \left[ \frac{Y(x) 1_{Y(x) - y^{\text{min}} < 0}}{E \left[ 1_{Y(x) - y^{\text{min}} < 0} | Y^{n,1} = y^{n,1} \right]} | Y^{n,1} = y^{n,1} \right] \right)
\]

\[
= P \left( Y(x) \not\geq y^{\text{min}} | Y^{n,1} = y^{n,1} \right) I \left( \overline{Y}(x) \right).
\]

A natural question one might ask is, what are the benefits of Method 2? Aside from its equivalence to the usual single-objective expected improvement function, there are three reasons it should be considered. First, it has made its way into the literature, as Keane (2006) essentially uses Method 2 to define \( QI(x) \) in a proposed expected improvement algorithm that will be presented shortly. Second, in situations where we must calculate expectations via Monte Carlo integration and \( I^*(\cdot) \) is expensive to compute, using Method 1 can be computationally demanding because \( I^*(\cdot) \) must be computed for every single draw from the predictive distribution. For example, the improvement functions defined by Emmerich et al. (2006) require a calculation of the hypervolume indicator which has a very high computational cost for larger \( m \) according to Fonseca et al. (2006) and Zitzler et al. (2008). Finally, the most compelling reason to study Method 2 is that it can work surprisingly well in practice and, when paired with certain improvement functions, can even outperform Method 1.
The choice of improvement function has not been resolved theoretically; several such functions have been proposed in the literature. In Section 2.2 we discuss several proposed approaches from the literature, and in Section 2.3 we introduce several new improvement functions. We discuss some of the theoretical properties of these improvement functions in Section 2.4. In Chapter 3, we compare the performance of the various improvement functions on real-world applications and test functions with Method 1 and Method 2 implementations with independent and nonseparably dependent multivariate Gaussian process models.

To keep notation simple, we will use the simplified expressions QIM1 and QIM2 for Method 1 and Method 2, respectively, throughout the rest of the thesis.

2.1.2 Handling Constraints in Multiobjective Optimization Problems

Schonlau (1997), Schonlau et al. (1998), and Williams et al. (2010) address the issue of single-objective optimization when there are constraint functions present in addition to the function to be minimized. Keane (2006) and Emmerich et al. (2006) investigate constrained multiobjective optimization in the context of their proposed methods. Here is the expected improvement approach to constrained multiobjective optimization. Suppose that, in addition to the \( m \) objective functions \( \mathbf{y}(\cdot) = y_1(\cdot), \ldots, y_m(\cdot) \), there are \( k \) computable constraint functions

\[
\mathbf{y}_c(\cdot) = (y_{m+1}(\cdot), \ldots, y_{m+k}(\cdot))
\]

that must satisfy \( l_i \leq y_{m+i}(\cdot) \leq u_i \) for each \( i \). To solve the constrained problem with an improvement function \( I^*(\cdot) \), we must use a modified improvement function

\[
I^*_c(\mathbf{y}(\mathbf{x}), \mathbf{y}_c(\mathbf{x})) = I^*(\mathbf{y}(\mathbf{x})) \prod_{i=1}^{k} \mathbf{1}_{[l_i \leq y_{m+i}(\mathbf{x}) \leq u_i]}
\]
which forces inputs that violate constraints to yield zero improvement. Also, after evaluating all \( m + k \) functions at our initial design \( D_n \), our output data consists not only of \( y_{m,n} \), but also
\[
y_c^{m,n} = (y_c^\top(x_1), \ldots, y_c^\top(x_n))^\top,
\]
which is the outputs of the constraints function at the initial design. Even though all data points will be used to build our statistical models, we must exclude infeasible points from \( P^o_n \). In our sequential design algorithm, we replace \( QI(x) \) with
\[
QI_c(x) = E \left[ I^* (Y(x)) \prod_{i=1}^k 1[l_i \leq y_{m+i} \leq u_i] | Y^{m,n} = y_{m,n}, Y_{c}^{m,n} = y_{c}^{m,n} \right]
\]
when using QIM1. When using QIM2, we have
\[
QI_c(x) = P \left( Y(x) \in \mathcal{R}_n, \prod_{i=1}^k l_i \leq Y_{m+i}(x) \leq u_i | Y^{m,n} = y_{m,n}, Y_{c}^{m,n} = y_{c}^{m,n} \right)
\times I^* (Y(x)_c)
\]
where
\[
Y(x)_c = E \left[ Y(x)_1 \begin{bmatrix} Y(x) & Y^{m,n} = y_{m,n}, Y_{c}^{m,n} = y_{c}^{m,n} \end{bmatrix} \right] \left| Y(x) \in \mathcal{R}_n, \prod_{i=1}^k l_i \leq Y_{m+i}(x) \leq u_i \right].
\]

With the exception of Williams et al. (2010), constrained optimization for single-objective optimization has been accomplished under the assumption that the objective and constraint functions are independent. In the multiobjective setting, all models from the current literature have assumed mutual independence between the set of constraint functions and the set of objective functions. The reason for doing so is
that the formulation of $QI_c(x)$ becomes much simpler. When using QIM1, we have

$$QI_c(x) = E[I^*(Y(x))|Y^{m,n} = y^{m,n}] \prod_{i=1}^{k} P(l_i \leq Y_{m+i}(x) \leq u_i|y_{m+i}(x_1), \ldots, y_{m+i}(x_n)).$$

Using QIM2, we have

$$QI_c(x) = I^*(\bar{Y}(x))P(Y(x) \in \mathcal{R}_n|Y^{m,n} = y^{m,n})$$

$$\times \prod_{i=1}^{k} P(l_i \leq Y_{m+i}(x) \leq u_i|y_{m+i}(x_1), \ldots, y_{m+i}(x_n)).$$

Not only do we not have to fit a cross-covariance here between the functions, but we can compute the expected improvement as a product of the unconstrained expected improvement and the probability that $x$ yields a feasible output. Moreover, this feasibility probability is just the product of $m$ univariate normal probabilities and is therefore quick to compute.

2.2 Improvement Functions from the Literature

2.2.1 0-1 Improvement

Keane (2006) introduced the simplest of all multiobjective expected improvement algorithms, which we will refer to as the probability of improvement. Essentially, we sequentially add input by maximizing $P(Y(x) \in \mathcal{R}_n|Y^{m,n} = y^{m,n})$. Here, we are using the improvement function

$$I_{PI}(y(x)) = 1_{[y(x) \in \mathcal{R}_n]}$$

We will refer to $I_{PI}(\cdot)$ as the 0-1 improvement function, as it returns a 0 if $y(x)$ is dominated by the current Pareto front, and a 1 otherwise. Notice that, for this particular improvement function, QIM1 and QIM2 are equivalent. For the case where $m = 2$ and the outputs are modeled as independent Gaussian processes, Keane (2006) derives a closed-form expression for $P(Y(x) \in \mathcal{R}_n|Y^{m,n} = y^{m,n})$. 

57
While relatively simple, the probability of improvement is not a very effective expected improvement function. As Keane (2006), Forrester et al. (2008), and Bautista (2009) have all pointed out, based on numerical studies, the probability of improvement tends to add outputs that are “clumped” together. Essentially, this particular choice for $QI(x)$ does not explore the input space very efficiently. This is not surprising, since the improvement function $I_{PI}(Y(x))$ only takes into account whether or not an improvement occurs and ignores the magnitude of the improvement.

**Implementation**

For an arbitrary output dimension $m$, one can employ a sample average approximation for

$$ QI(x) = P(Y(x) \in R_n | Y^{m,n} = y^{m,n}). $$

To do so, first generate a random sample

$$ Z^1, \ldots, Z^S \overset{iid}{\sim} N(0_m, I_m). $$

Then, for any $x$, letting $C(x)$ be the Cholesky decomposition of $S(x)$ (the covariance of $Y(x)|Y^{m,n} = y^{m,n}$), we can transform each $Z^j$ into a random variable

$$ Y^j(x) = C(x)Z^j + \hat{y}(x) \sim N(\hat{y}(x), S(x)) , \quad 1 \leq j \leq S. $$

We can use these samples to obtain an approximation

$$ \hat{QI}(x) = \frac{1}{S} \sum_{j=1}^{S} 1[Y^j(x) \in R_n], $$

which is a deterministic function for a given sample $Z^1, \ldots, Z^S$. We then apply a direct search optimization algorithm to $\hat{QI}(x)$. 58
2.2.2 Keane’s Distance-Based Improvement

Keane (2006) also introduced a distance-based expected improvement algorithm in the case where \( m = 2 \), with the goal of balancing exploration of the input space and exploitation of the surrogate approximation in the search for nondominated sets. This approach employs a QIM2 based \( QI(x) \). While only presented for two-dimensional outputs, it is not difficult (at least in principle) to generalize for arbitrary \( m \geq 2 \).

Here, we have

\[
QI(x) \equiv P(Y(x) \in R_n|Y_{m,n}) \times \min_{x_i \in P^X_n} \sqrt{\sum_{k=1}^{m} (Y_k(x) - y_k(x_i))^2} = \min_{x_i \in P^X_n} \left[ \sum_{k=1}^{m} \left( E \left[ (Y_k(x) - y_k(x_i)) 1[Y(x) \in R_n|Y_{m,n}] \right] \right)^2 \right] \tag{2.2.1}
\]

The improvement function implicit in this formulation is

\[
I_K(y(x)) = \begin{cases} 
\min_{x_i \in P^X_n} \sqrt{\sum_{k=1}^{m} (y_k(x) - y_k(x_i))^2} & \text{if } y(x) \in R_n, \\
0 & \text{otherwise.}
\end{cases}
\]

Just like \( P(Y(x) \in R_n|Y_{m,n} = y^{m,n}) \), Keane (2006) derives a closed form expression for \( Y(x) \) in the biobjective, independent-output case. A plot of \( I_K(y(x)) \) corresponding to a four-point current Pareto front is shown Figure 2.1.

This expected improvement function can substantially outperform the probability of improvement, as the distance-based improvement function 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 points to be more spread out in the objective space than when using the probability of improvement. However, \( I_K(y(x)) \) has some theoretical disadvantages that will be presented
Figure 2.1: Surface plot of $I_K(y(x))$ when $m = 2$. The black squares represent the current Pareto front.

in Section 2.4.2. Also, $I_K(y(x))$ is relatively inefficient when compared to most other proposed methods, as we will exhibit in Chapter 3.

QIM1 - Implementation

For any $m \geq 2$, we optimize

$$QI(x) = E[I_K(Y(x)) | Y^{m,n} = y^{m,n}]$$

via sample average approximation (SAA), as described in Shapiro (2003). The idea is to construct an approximation to $E[I_K(Y(x)) | Y^{m,n} = y^{m,n}]$ based on a random
sample from the distribution of $Y(x)$, and then optimize this easy-to-calculate approximation. To implement an SAA, we first generate a random sample

$$Z^1, \ldots, Z^S \overset{iid}{\sim} N(0_m, I_m).$$

(2.2.2)

For any given $x$, letting $C(x)$ be the Cholesky decomposition of $S(x)$ (the covariance of $Y(x)|Y^{m,n} = y^{m,n}$), we can transform each $Z^i$ into a random variable

$$Y^i(x) = C(X)Z^i + \hat{y}(x) \sim N(\hat{y}(x), S(x)).$$

(2.2.3)

Thus, $Y^1(x), \ldots, Y^S(X)$ is a sample from the distribution of $Y(x)|Y^{m,n} = y^{m,n}$. Letting

$$\hat{Q}_I(x) = \frac{1}{S} \sum_{s=1}^{S} I_K(Y^s(x))$$

be the so-called sample average function, we then find our next input by calculating

$$x^{n+1} = \arg \max \hat{Q}_I(x) = \arg \max \frac{1}{S} \sum_{s=1}^{S} I_K(Y^s(x))$$

via a direct search optimization algorithm.

**QIM2 - Implementation**

In this case, $Q_I(x)$ is defined as

$$Q_I(x) = P(Y(x) \in R_n|Y^{m,n} = y^{m,n}) I_K(Y(x)).$$

Even though $Q_I(x)$ is not an expectation for QIM2, for an arbitrary output dimension $m$, we can still set up an approximation for $Q_I(x)$ where all random samples are drawn before running the deterministic direct search optimization algorithm. Such an approach is similar in spirit to the sample average approximations used for the QIM1 implementation. First, generate a random sample

$$Z^1, \ldots, Z^S \overset{iid}{\sim} N(0_m, I_m).$$

(2.2.4)
Then, for any \( x \), letting \( C(x) \) be the Cholesky decomposition of \( S(x) \) (the covariance of \( Y(x)|Y^{m,n} = y^{m,n} \)), we can transform each \( Z^j \) into a random variable

\[
Y^j(x) = C(x)Z^j + \hat{y}(x) \sim N(\hat{y}(x), S(x)). \tag{2.2.5}
\]

So, the \( Y^j(x) \)'s are draws from the posterior predictive distribution at \( x \). We can use these samples to obtain the following approximations to \( P(Y(x) \in \mathcal{R}_n|Y^{m,n} = y^{m,n}) \) and \( \bar{Y}(x) \):

\[
\hat{P}(Y(x) \in \mathcal{R}_n|Y^{m,n} = y^{m,n}) = \frac{1}{S} \sum_{j=1}^{S} 1[Y^j(x) \in \mathcal{R}_n] \tag{2.2.6}
\]

\[
\bar{Y}(x) = \frac{\sum_{j=1}^{S} Y^j(x) 1[Y^j(x) \in \mathcal{R}_n]}{\sum_{j=1}^{S} 1[Y^j(x) \in \mathcal{R}_n]} \tag{2.2.7}
\]

We can then approximate \( QI(x) \) as

\[
\hat{QI}(x) = \hat{P}(Y(x) \in \mathcal{R}_n|Y^{m,n} = y^{m,n}) I_K \left( \bar{Y}(x) \right).
\]

Similar to the sample average approximation used in QIM1, \( \hat{QI}(x) \) is a deterministic function for a particular realization of the random sample \( Z^1, \ldots, Z^S \).

### 2.2.3 Hypervolume Improvement

Emmerich et al. (2006) incorporated the hypervolume indicator into an improvement function. Refer to (1.5.1) for a definition of the hypervolume indicator. The idea is to choose the next point \( x \) in our sequential design so that we get the largest possible expected increase in the hypervolume indicator. To do this, we define the hypervolume improvement function as

\[
I_H(y(x)) = \begin{cases} 
0 & \text{if } y(x) \preceq P_{Y_0}^n \text{ or } y(x) \npreceq R \\
I_H \left( \{y(x)\} \cup P_{Y_0}^n, R \right) - I_H \left( P_{Y_0}^n, R \right) & \text{otherwise}
\end{cases}
\]
A plot of $I_H(y(x))$, for the same four point current Pareto front, can be seen in Figure 2.2. Notice that evaluation of the hypervolume requires some knowledge of the objective space, as we must come up with a suitable reference point $R$. The hypervolume improvement function has many desirable theoretical properties that will be explored in Chapter 2, and it performs very well on real world problems. However, the inherent computational demands associated with the hypervolume indicator do lead to some practical concerns, especially as the objective space dimension $m$ become larger.

Figure 2.2: Surface plot of $I_H(y)$ when $m = 2$ and $R = (0.9, 0.9)$. The black squares represent the current Pareto front.
It should be noted that Emmerich et al. (2006) did not use the expected hypervolume improvement in the general expected improvement framework. Instead, it was used as a pre-screening procedure for a multiobjective evolutionary algorithm (MOEA). At each iteration of this MOEA, a subset of the current inputs where the functions $y(\cdot)$ have been evaluated were selected for the usual genetic operations of crossover and mutation. Standard MOEAs would require that $y(\cdot)$ be evaluated at all new inputs created by these genetic operations. Rather than evaluating $y(\cdot)$ at all of these new candidate inputs, $E[I_H(Y(x))|Y^{m,n} = y^{m,n}]$ was calculated at all of these new inputs, and $y(\cdot)$ was only evaluated at a subset that contained the highest expected hypervolume improvements. High computational costs associated in evaluating this QIM1 expected improvement (which are discussed in Section 2.4) were not problematic here because the expected improvement only needed to be evaluated for a small, finite set of inputs. However, when using it in the previously presented expected improvement framework, the expected improvement function needs to be optimized using some sort of direct search algorithm, which will require calculations of $E[I_H(Y(x))|Y^{m,n} = y^{m,n}]$ for a large number of $x$’s.

**QIM1 - Implementation**

When the output dimension $m$ is small (such as $m=2$), the QIM1 implementation is rather straightforward. For any given $x$, we can construct a sample $Y^1(x), \ldots, Y^S(x)$ from the posterior predictive distribution of $Y(x)$ from an initial random sample $Z^1, \ldots, Z^S$, as described in (2.2.2) - (2.2.3). We then let

$$\hat{QI}(x) = \frac{1}{S} \sum_{s=1}^{S} I_H(Y^s(x))$$
be our sample average function, which is optimized via a direct search optimization algorithm. Notice that such an implementation potentially requires the calculation of $S$ hypervolumes. As long as $m$ is small, this is not a problem, as hypervolume can be computed quickly and exactly.

However, when the output dimension is even moderately sized, we must also rely on an approximation to calculate each $I_H(Y^i(x))$ since exact hypervolume calculations are too computationally expensive to compute. To do so, we must first approximate $I_H(P^n_{Y^i}, R)$. This does not depend on $x$, so this calculation can be done before starting the direct search optimization algorithm. Then, for any given $x$, we need to approximate $I_H \left( \{ Y^i(x) \} \cup P^n_{Y^i}, R \right)$ for any $Y^i(x) \in \mathcal{R}_n$.

To approximate $I_H \left( P^n_{Y^i}, R \right)$, first generate a random sample

$$U_1, \ldots, U_T \overset{iid}{\sim} \text{Uniform} ([0, 1]^m). \quad (2.2.8)$$

Let $Y^{min} = [Y^{min}_1, \ldots, Y^{min}_m]$ be an $m$ dimensional vector comprised of the smallest value in each dimension of $P^n_{Y^i}$. Then, calculate the total volume in the $m$ dimensional cube bounded by $Y^{min}$ and $R$ as

$$Vol_T = \prod_{i=1}^{m} ([R]_i - [Y^{min}]_i). \quad (2.2.9)$$

Then, let $P^n_{Y^i*}$ be a transformed version of $P^n_{Y^i}$ defined as

$$P^n_{Y^i*} = (P^n_{Y^i} - 1_p \otimes Y^{min}) \text{diag} \left( \frac{1}{[R]_1 - [Y^{min}]_1}, \ldots, \frac{1}{[R]_m - [Y^{min}]_m} \right) \quad (2.2.10)$$

where $p$ is the number of points in $P^n_{Y^i}$. Then, we can use

$$I_H \left( P^n_{Y^i}, R \right) = Vol_T \times \frac{1}{T} \sum_{t=1}^{T} 1_{[P^n_{Y^i*} \geq U_t]} \quad (2.2.11)$$

as an approximation to $I_H \left( P^n_{Y^i}, R \right)$. To approximate $I_H \left( \{ Y^i(x) \} \cup P^n_{Y^i}, R \right)$ when $Y^i(x) \in \mathcal{R}_n$ and $Y^i(x) \succ R$ we can proceed in a similar fashion. Let $Y^{min}(x)$ be an
Then, calculate the total volume in the $m$ dimensional cube bounded by $Y^{min}(x)$ and $R$ as

$$Vol_T(x) = \prod_{i=1}^{m} (|R|_i - |Y^{min}(x)|_i).$$

(2.2.12)

Then, let $P^*_n$ be a transformed version of $P^n$ defined as

$$P^*_{n+1}(x) = (P^*_{n+1}(x) - 1_p(x) \otimes Y^{min}(x))$$

$$\times \text{diag} \left( \frac{1}{|R|_1 - |Y^{min}(x)|_1}, \ldots, \frac{1}{|R|_m - |Y^{min}(x)|_m} \right)$$

(2.2.13)

where $p(x)$ is the number of points in $P_{Y^*}^{n+1}(x)$. Then, we can use

$$\widehat{I}_H (P^*_{Y^*}^{n+1}(x), R) = Vol_T(x) \times \frac{1}{T} \sum_{t=1}^{T} 1_{[(P^*_{n+1}(x)) \geq U_t]}$$

(2.2.14)

as an approximation to $I_H (P^*_{Y^*}^{n+1}(x), R)$.

We can now construct a sample average approximation of the posterior expected hypervolume improvement. To do so, let

$$h(Y^n(x)) = \left\{\begin{array}{ll}
0 & \text{if } Y^n(x) \subseteq P^*_n \text{ or } Y^n(x) \not\supseteq R \\
\widehat{I}_H (P^*_{Y^*}^{n+1}(x), R) - \widehat{I}_H (P^n, R) & \text{otherwise.}
\end{array}\right.$$  

We can then approximate $QI(x)$ as

$$\overline{QI(x)} = \frac{1}{S} \sum_{s=1}^{S} h(Y^n(x))$$

which is a deterministic function for a particular realization of $U_1, \ldots, U_T$ and $Z^1, \ldots, Z^S$.

**QIM2 - Implementation**

In this case, $QI(x)$ is defined as

$$QI(x) = P(Y(x) \in \mathcal{R}_n | \mathcal{Y}^{m,n} = y^{m,n}) I_H (\mathcal{Y}(x)).$$
Even though $QI(x)$ is not an expectation in the QIM2 setting, we can still approximate $QI(x)$ based on random samples that are drawn before running a deterministic direct search optimization algorithm. To do so, first construct an approximation for $P(Y(x) \in \mathcal{R}_n | Y^{m,n} = y^{m,n})$ and $\overline{Y}(x)$, as described in (2.2.4)-(2.2.7). Approximate $QI(x)$ by

$$\hat{QI}(x) = \hat{P}(Y(x) \in \mathcal{R}_n | Y^{m,n} = y^{m,n}) I_H(\overline{Y}(x)) \quad (2.2.15)$$

As long as $m$ is small, we can get exact calculations of $I_H(\overline{Y}(x))$ with little difficulty. The advantage of this approach over the QIM1 approach in this setting is obvious; only one hypervolume calculation is needed at each $x$, rather than potentially $S$ hypervolume calculations.

For larger $m$, we may still need to calculate the hypervolume improvement function via a Monte Carlo approximation. To do so, generate a uniform sample over $[0, 1]^m$ just as in (2.2.8). Use this sample to obtain $\hat{I}_H(P^n Y, R)$ as in (2.2.11). Then, let $P^{n+1}_Y(\overline{Y}(x))$ be the current Pareto front if $P^n_Y$ is augmented by $\overline{Y}(x)$, and let $\overline{Y}^{\text{min}}(x)$ be an $m$ dimensional vector comprised of the smallest value in each dimension of $P^{n+1}_Y(\overline{Y}(x))$. Calculate the total volume in the $m$ dimensional cube bounded by $\overline{Y}^{\text{min}}(x)$ and $R$ as

$$Vol_T(\overline{Y}(x)) = \prod_{i=1}^m \left( [R]_i - \left[ \overline{Y}^{\text{min}}(x) \right]_i \right). \quad (2.2.16)$$

Then let $P^{n+1}_Y(\overline{Y}(x))$ be the transformed version of $P^{n+1}_Y(\overline{Y}(x))$ defined as

$$P_{Y^*}^{n+1}(\overline{Y}(x)) = \left( P^{n+1}_Y(\overline{Y}(x)) - 1_p(\overline{Y}(x)) \otimes \overline{Y}^{\text{min}}(x) \right)$$
\[
\times \text{diag}
\left(\frac{1}{[R]_1 - \hat{Y}^{\text{min}}(x)}, \ldots, \frac{1}{[R]_m - \hat{Y}^{\text{min}}(x)}\right)
\]

where \(p\left(\hat{Y}(x)\right)\) is the number of points in \(\mathcal{P}^{n+1}_y\left(\hat{Y}(x)\right)\). Then, we can approximate \(I_H\left(\hat{Y}(x)\right)\) in (2.2.15) by

\[
\hat{I}_H\left(\hat{Y}(x)\right) = \begin{cases} 
\text{Vol}_T\left(\hat{Y}(x)\right) \times \frac{1}{T} \sum_{t=1}^{T} 1_{\mathcal{P}^{n+1}_y\left(\hat{Y}(x)\right) \succeq \{U_t\}} - \hat{I}_H\left(\mathcal{P}^n_y, R\right) & \text{if } \hat{Y}(x) \succ R \\
0 & \text{otherwise.}
\end{cases}
\]

### 2.2.4 Pareto Improvement

Bautista (2009) introduced the \(EmaX\) algorithm, which follows the general framework outlined at the start of this subsection with the *Pareto improvement function* and the QIM1 interpretation of \(QI(x)\). The Pareto improvement function is explicitly defined as

\[
I_P\left(y(x)\right) = -\max_{x_i \in \mathcal{P}^n_y} \min_{j=1, \ldots, m} (y_j(x) - y_j(x_i))
\]

Figure 2.3 displays in example of \(I_P(\cdot)\) when \(m = 2\) for the same four point current Pareto front 2.2.

Bautista (2009) shows that \(I_P\left(y(x)\right)\) is an attractive improvement function because it allows one to easily distinguish whether or not a given point in the objective space is nondominated. If \(I_P\left(y(x)\right) > 0\), then \(y(x)\) is not dominated by any vectors in \(\mathcal{P}^n_y\). If \(I_P\left(y(x)\right) < 0\), then \(y(x)\) is dominated by at least one vector in \(\mathcal{P}^n_y\). If \(I_P\left(y(x)\right) = 0\), then \(y(x)\) is weakly dominated by at least one vector in \(\mathcal{P}^n_y\). This improvement function is based on the modified maximin fitness function

\[
\text{fitness}(y(x)) = \max_{x_i \in \mathcal{P}^n_x} \min_{j=1, \ldots, m} (y_j(x) - y_j(x_i))
\]  

(2.2.17)
Figure 2.3: Surface plot of $I_P(y(x))$ when $m = 2$. The black squares represent the current Pareto front.

presented in Balling (2003), which was originally was introduced as a component of a multiobjective evolutionary algorithm. One can interpret the Pareto improvement (and the closely-related maximin fitness function) as a distance metric from an objective space vector to the Pareto front.

QIM1 - Implementation

For arbitrary $m$, we optimize

$$QI(x) = E[I_P(Y(x)) | Y^{m,n} = y^{m,n}]$$

via sample average approximation (SAA), as described in Shapiro (2003). For any given $x$, we can construct a sample $Y^1(x), \ldots, Y^S(x)$ from the posterior predictive
distribution of $\mathbf{Y}(\mathbf{x})$ from an initial random sample $\mathbf{Z}^1, \ldots, \mathbf{Z}^S$, as described in (2.2.2) - (2.2.3). Letting

$$
\widehat{Q}_I(\mathbf{x}) = \frac{1}{S} \sum_{s=1}^{S} I_P(Y^s(\mathbf{x}))
$$

(this is the sample average function and it is a deterministic function for a particular realization of the random sample $\mathbf{Z}^1, \ldots, \mathbf{Z}^S$), we then find our next input by calculating

$$
\mathbf{x}^{n+1} = \arg \max \widehat{Q}_I(\mathbf{x}) = \arg \max \frac{1}{S} \sum_{s=1}^{S} I_P(Y^s(\mathbf{x}))
$$

via a direct search optimization algorithm.

**QIM2 - Implementation**

Discussion of a QIM2 implementation of the Pareto improvement function will be deferred to Section 2.3.1, since the QIM2 implementation of $I_P(\cdot)$ is identical to a newly proposed improvement function.

**2.3 Proposed Improvement Functions**

**2.3.1 Maximin Improvement**

In Section 2.2.4, we introduced the Pareto improvement function used in the $EmaX$ algorithm of Bautista (2009). A potential drawback of this particular improvement function, which will be discussed in Section 2.4, is that it is not equal to zero in the currently dominated region, and one consequence of this definition is that the improvement function for single-objective optimization employed in Jones et al. (1998) and Schonlau (1997) is not a special case of the Pareto improvement function.

70
This issue can be easily resolved by introducing the *maximin improvement function*, which we will define as

\[
I_M(y(x)) = - \max_{x_i \in P_X} \min_{j=1,\ldots,m} (y_j(x) - y_j(x_i)) \left[ - \max_{x_i \in P_X} \min_{j=1,\ldots,m} (y_j(x) - y_j(x_i)) < 0 \right]
\]

(Figure 2.4 illustrates the two-dimensional case with the same four point current Pareto front.) Essentially, this improvement function is identical to the Pareto improvement function in the currently nondominated region \( \mathcal{R}_n \) of the objective space, but is equal to zero in the entire dominated region.

![Surface plot of \( I_M(y(x)) \) when \( m = 2 \). The black squares represent the current Pareto front.](image)

Figure 2.4: Surface plot of \( I_M(y(x)) \) when \( m = 2 \). The black squares represent the current Pareto front.
There exists an interesting relationship between the maximin improvement function and the binary-$\epsilon$ indicator. To describe this relationship, first let $P_{Y}^{n+1}(y(x))$ be the current Pareto front if $y^{m,n}$ is augmented by $y(x)$. Then, one could think of $I_{\epsilon} (P_{Y}^{n}, P_{Y}^{n+1}(y(x)))$ as quantifying how much better $P_{Y}^{n+1}(y(x))$ is than $P_{Y}^{n}$, i.e., how much $y(x)$ improves upon our current best Pareto front approximation. It is reasonable, then, to use $I_{\epsilon} (P_{Y}^{n}, P_{Y}^{n+1}(y(x)))$ as an improvement function, replace $y(x)$ by $Y(x)$ in $I_{\epsilon} (P_{Y}^{n}, P_{Y}^{n+1}(y(x)))$ and (in the QIM1 setting) choose $x$ to maximize

$$E \{ I_{\epsilon} (P_{Y}^{n}, P_{Y}^{n+1}(x)) | Y^{m,n} = y^{m,n} \}.$$  

It can be shown that $I_{\epsilon} (P_{Y}^{n}, P_{Y}^{n+1}(x))$ and $I_{M} (y(x))$ are equivalent. Thus, using the maximin improvement function to control the search for the Pareto front is essentially equivalent to using the additive binary-$\epsilon$ indicator to control the search for the Pareto front. We formally state this relationship in the following theorem:

**Theorem 4.** Let $P_{Y}^{n+1}(y(x))$ be the set of nondominated points in the set $P_{Y}^{n} \cup \{y(x)\}$. Then,

$$I_{M} (y(x)) = I_{\epsilon} (P_{Y}^{n}, P_{Y}^{n+1}(y(x))).$$

**Proof.** See Appendix B.1. \qed

When using the maximin improvement function, the scaling of the various outputs matters, so we will empirically scale the outputs as

$$\frac{y_i(x) - \min \{y_i(x_1), \ldots, y_i(x_m)\}}{\max \{y_i(x_1), \ldots, y_i(x_m)\} - \min \{y_i(x_1), \ldots, y_i(x_m)\}}$$  \hspace{1cm} (2.3.1)

so that, for each objective $y_i(\cdot)$, $1 \leq i \leq m$, $\min \{y_i(x_1), \ldots, y_i(x_m)\} = 0$ and $\max \{y_i(x_1), \ldots, y_i(x_m)\} = 1.$
Implementation - QIM1

In the biobjective case, we can obtain a (nearly-analytic) expression for

$$E[I_M(Y(x)) | Y^{m,n} = y^{m,n}]$$

that can be implemented quite accurately. Let $P_X^n = \{x_1^*, \ldots, x_p^*\}$ and $P_Y^m = \{y(x_1^*), \ldots, y(x_p^*)\}$, $p \leq n$. Without loss of generality, assume that $y_1(x_1^*) \leq \ldots \leq y_1(x_p^*)$. As a consequence of the fact that $P_Y^m$ cannot contain any dominated points, we must then have $y_2(x_1^*) \geq \ldots \geq y_2(x_p^*)$. In the biobjective case, we can express the mean and covariance of the interpolating process $Y(x)$ by

$$\hat{y}(x) = \begin{bmatrix} \hat{y}_1(x) \\ \hat{y}_2(x) \end{bmatrix}$$

and

$$S(x) = \begin{bmatrix} s_1^2(x) & \rho(x)s_1(x)s_2(x) \\ \rho(x)s_1(x)s_2(x) & s_2^2(x) \end{bmatrix}.$$ 

respectively, where $\rho(x)$ is the correlation between the two outputs. For notational convenience, let $y_1(x_{p+1}) = y_2(x_0) = \infty$, $k(1) = 2$, $k(2) = 1$, $h(1,j) = j + 1$, and $h(2,j) = j - 1$.

$I_M(Y(x))$ partitions $\mathbb{R}^d$ into $2p + 1$ regions $R_{1,1}, \ldots, R_{1,p}, R_{2,1}, \ldots, R_{2,p}$ and $R_D$, where

$$R_{i,j} = \{(y_i, y_k(i)) : y_i \leq y_i(x_j^*), y_k(i)(x_j^*) - y_i(x_j^*) + y_i \leq y_k(i)(x_h(i,j)) - y_i(x_j^*) + y_i\}$$

and

$$R_D = \{(y_1, y_2) : \{y_1, y_2\} < P_Y^n\}. \quad (2.3.2)$$

73
Figure 2.5: Regions of integration for a three point Pareto front

(See Figure 2.5 for an example.) In $R_{i,j}$, $I_M(x)$ is equal to $y_i(x_j^*) - y_i$, and in $R_D$, $I_M(x)$ is equal to 0. Therefore, If we let

$$\text{Int}_{i,j} = \int_{y_i(x_j^*)}^{y_i(x_j^*)} \int_{y_k(x_i^*)}^{y_k(x_i^*)} [y_i(x_j^*) - y_i] \cdot \text{f}(y_1, y_2) \, dy_k \, dy_i$$

(2.3.3)

where $i = 1, 2, j = 1, \ldots, p$, and $f(y_1, y_2)$ is the bivariate normal probability density function associated with the distribution of $Y(x)$, we have

$$E[I_M(x)] = \sum_i \sum_j \text{Int}_{i,j}.$$  

(2.3.4)

We can then prove the following theorem:
Theorem 5.

\[ E [I_M (Y (x)) | Y^{m,n} = y^{m,n}] = \sum_{i=1}^{2} \sum_{j=1}^{p} (Int_{i,j}^1 (x) + Int_{i,j}^2 (x) + Int_{i,j}^3 (x)) \quad (2.3.5) \]

where

\[ Int_{i,j}^1 (x) = s_i (x) \phi \left( \frac{d(i, j)}{s_i (x)} \right) \]

\[ \times \left[ \Phi \left( \frac{-d(k(i), j) + \rho (x) s_{k(i)} (x) d(i, j)/s_i (x)}{\sqrt{(1 - \rho^2 (x)) s_{k(i)}^2 (x)}} \right) \right. \]

\[ - \Phi \left( \frac{-d(k(i), h(i, j)) + \rho (x) s_{k(i)} (x) d(i, j)/s_i (x)}{\sqrt{(1 - \rho^2 (x)) s_{k(i)}^2 (x)}} \right) \right] , \]

\[ Int_{i,j}^2 (x) = \sqrt{A(i, j)} \frac{s_i (x) - s_{k(i)} (x) \rho (x)}{\sqrt{2\pi (1 - \rho^2 (x)) s_{k(i)}^2 (x)}} \]

\[ \times \left[ \exp \left\{ -\frac{1}{2} \left[ \frac{\hat{y}_{i}^2 (x)}{s_i^2 (x)} + \frac{(y_i (x_i^*) - d(k(i), j) + \rho (x) s_{k(i)} (x) \tilde{y}_i (x)/s_i (x))^2}{\sqrt{(1 - \rho^2 (x)) s_{k(i)}^2 (x)}} \right] \right\} \right. \]

\[ \times \exp \left\{ \frac{1}{2} A(i, j) v^2 (i, j) \right\} \Phi \left( \frac{y_i (x_i^*) - A(i, j) v(i, j)}{\sqrt{A(i, j)}} \right) \]

\[ - \exp \left\{ -\frac{1}{2} \left[ \frac{\hat{y}_{i}^2 (x)}{s_i^2 (x)} + \frac{(y_i (x_i^*) - d(k(i), h(i, j)) + \rho (x) s_{k(i)} (x) \tilde{y}_i (x)/s_i (x))^2}{\sqrt{(1 - \rho^2 (x)) s_{k(i)}^2 (x)}} \right] \right\} \]

\[ \times \exp \left\{ \frac{1}{2} A(i, j) v^2 (i, h(i, j)) \right\} \Phi \left( \frac{y_i (x_i^*) - A(i, j) v(i, h(i, j))}{\sqrt{A(i, j)}} \right) \]

\[ Int_{i,j}^3 (x) = (y_i (x_i^*) - \hat{y}_i (x)) \]

\[ \times \left[ \int_0^{u(i,j)} \Phi \left( \frac{d(i, j) - d(k(i), j) + (s_{k(i)} (x) \rho (x) - s_i (x)) \Phi^{-1} (w)}{\sqrt{(1 - \rho^2 (x)) s_{k(i)}^2 (x)}} \right) dw \right] \]

\[ - \int_0^{u(i,j)} \Phi \left( \frac{d(i, j) - d(k(i), h(i, j)) + (s_{k(i)} (x) \rho (x) - s_i (x)) \Phi^{-1} (w)}{\sqrt{(1 - \rho^2 (x)) s_{k(i)}^2 (x)}} \right) dw \]
In the above expressions,

\[ u(i, j) = \Phi \left( \frac{d(i, j)}{s_i(x)} \right), \]

\[ v(i, j) = \frac{\hat{y}_i(x)}{s_i^2(x)} + \frac{y_2(x_j^*) - d(k(i), j) + \rho(x) s_{k(i)}(x) \hat{y}_i(x)}{\sqrt{(1 - \rho^2(x)) s_{k(i)}^2(x)}}, \]

\[ A(i, j) = \frac{(1 - \rho^2(x)) s_{k(i)}^2(x) s_i^2(x)}{s_i^2(x) + s_{k(i)}^2(x) - 2\rho(x) s_i(x) s_{k(i)}(x)} \]

and

\[ d(i, j) = y_i(x_j^*) - \hat{y}_i(x). \]

Proof. The theorem is proved in Appendix B.2 by showing that \( \text{Int}_{i,j} = \text{Int}_{1,i,j} + \text{Int}_{2,i,j} + \text{Int}_{3,i,j} \).

Therefore, when \( m = 2 \), \( E[I_M(Y(x)) | Y_{m,n} = y_{m,n}] \) can be directly calculated and is optimized in a straightforward manner. The only minor difficulty is that \( \text{Int}_{3,i,j}(x) \) must be calculated via numerical quadrature. This numerical calculation is deterministic, however, and no Monte Carlo methods are needed.

For \( m > 2 \), we optimize

\[ QI(x) = E[I_M(Y(x)) | Y_{m,n} = y_{m,n}] \]

via sample average approximation (SAA), as described in Shapiro (2003). For any given \( x \), we can construct a sample \( Y^1(x), \ldots, Y^S(x) \) from the posterior predictive distribution of \( Y(x) \) from an initial random sample \( Z^1, \ldots, Z^S \), as described in (2.2.2) - (2.2.3). Letting

\[ \hat{QI}(x) = \frac{1}{S} \sum_{s=1}^{S} I_M(Y^s(x)) \]
(this is the sample average function and it is a deterministic function for a particular realization of the random sample $Z^1, \ldots, Z^S$), we then find our next input by calculating

$$x^{n+1} = \arg \max \widehat{QI}(x) = \arg \max \frac{1}{S} \sum_{s=1}^{S} I_M(Y^s(x))$$

via a direct search optimization algorithm.

**Implementation - QIM2**

In this case, $QI(x)$ is defined as

$$QI(x) = P(Y(x) \in R_n | Y^{m,n} = y^{m,n}) I_M(Y(x)) .$$

Even though $QI(x)$ is not an expectation in the QIM2 sense, we can still set up an approximation to $QI(x)$ where all random sample are drawn before running the deterministic direct search optimization algorithm. To do so, first construct approximation for $P(Y(x) \in R_n | Y^{m,n} = y^{m,n})$ and $\overline{Y}(x)$, as described in (2.2.4)-(2.2.7). We can then approximate $QI(x)$ as

$$\widehat{QI(x)} = \widehat{P}(Y(x) \in R_n | Y^{m,n} = y^{m,n}) I_M(\overline{Y}(x))$$

Just like when using QIM1, $\widehat{QI(x)}$ is a deterministic function for a particular realization of an initial $N(0_m, I_m)$ random sample $Z^1, \ldots, Z^S$.

Notice that, since $\overline{Y}(x) \in R_n$ by construction, $I_P(\overline{Y}(x)) = I_M(\overline{Y}(x))$. This is why we only discussed a QIM1 implementation of the Pareto improvement function. A QIM2 implementation of the Pareto improvement function would be equivalent to using the $QI(x)$ developed in the current subsection.
2.3.2 Gaussian Weighted Hypervolume Improvement

To define the \textit{Gaussian weighted hypervolume improvement}, first empirically scale the data as in (2.3.1). Recall that the weighted hypervolume indicator (1.5.1) depends on a weight function \( w(\cdot) \). Letting \( w(y) = \prod_{i=1}^{m} \frac{1}{\zeta} \phi \left( \frac{y_i}{\zeta} \right) \), we define the \textit{Gaussian weighted hypervolume improvement} as

\[
I_{WH}(y(x)) = I_H^w (P_{y}^{n+1}(y(x))) - I_H^w (P_y^n)
= \int_{\mathbb{R}^m} \prod_{i=1}^{m} \frac{1}{\zeta} \phi \left( \frac{y_i}{\zeta} \right) 1_{[P_{y}^{n+1}(y(x)) \succeq \{y\}]} dy
- \int_{\mathbb{R}^m} \prod_{i=1}^{m} \frac{1}{\zeta} \phi \left( \frac{y_i}{\zeta} \right) 1_{[P_y^n \succeq \{y\}]} dy. \tag{2.3.6}
\]

The following theorem provides us with a more computationally friendly form for \( I_{WH}(y(x)) \).

\textbf{Theorem 6.} Let \( W \) be a random vector in \( \mathbb{R}^m \). Then,

\[
P \left( P_{y}^{n+1}(y(x)) \succeq W \right) - P \left( P_y^n \succeq W \right) = P \left( y(x) \succeq W, W \in \mathcal{R}_n \right), \tag{2.3.7}
\]

where \( P_y^n \) is defined in Section 2.1.1 and \( \mathcal{R}_n \) is defined in (2.1.3).

\textit{Proof.} Consider two cases.

- \textbf{Case 1:} \( y(x) \in \mathcal{R}_n \)

  First, notice that we can express the left hand side of (2.3.7) as

  \[
  E \left[ 1_{[P_{y}^{n+1}(y(x)) \succeq W]} - 1_{[P_y^n \succeq W]} \right]. \tag{2.3.8}
  \]

  Notice that \( 1_{[P_{y}^{n+1}(y(x)) \succeq W]} - 1_{[P_y^n \succeq W]} \) is equal to 1 if and only if \( P_{y}^{n+1}(y(x)) \succeq W \) and \( W \in \mathcal{R}_n \). If \( W \in \mathcal{R}_n \), then \( W \not\succeq y \) for all \( y \in P_y^n \). The vector
\( y(x) \) is the lone element in \( \mathcal{P}_{Y}^{n+1}(y(x)) \) that is not in \( \mathcal{P}_{Y}^{n} \), so \( W \in \mathcal{R}_{n} \) and \( \mathcal{P}_{Y}^{n+1}(y(x)) \succeq W \) if and only if \( W \preceq y(x) \) and \( W \in \mathcal{R}_{n} \). Therefore,

\[
1_{\mathcal{P}_{Y}^{n+1}(y(x)) \succeq W} - 1_{\mathcal{P}_{Y}^{n} \succeq W} = 1_{[y(x) \succeq W, W \in \mathcal{R}_{n}]}.
\]

Taking expectations with respect to \( W \), we can conclude that

\[
P (\mathcal{P}_{Y}^{n+1}(y(x)) \succeq W) - P (\mathcal{P}_{Y}^{n} \succeq W) = P (y(x) \succeq W, W \in \mathcal{R}_{n})
\]

- **Case 2**: \( y(x) \not\in \mathcal{R}_{n} \)

In this case, \( \mathcal{P}_{Y}^{n+1}(y(x)) = \mathcal{P}_{Y}^{n} \), so the left hand side of (2.3.7) is zero. Additionally, the events \( W \preceq y(x) \) and \( W \in \mathcal{R}_{n} \) cannot simultaneously occur, so \( P (y(x) \succeq W, W \in \mathcal{R}_{n}) = 0 \) as well.

Notice that the theorem applies to *any* random vector \( W \) in the objective space, not just one with a joint pdf specified by \( w(y) \). This fact makes the theorem useful for the completeness indicator improvement function introduced in Section 2.3.3.

Applying Theorem 6 to (2.3.6) with \( V \sim \mathcal{N}(0_{m}, \zeta^{2}I_{m}) \) playing the role of \( W \), we have

\[
I_{WH}(y(x)) = P (\mathcal{P}_{Y}^{n+1}(y(x)) \succeq V) - P (\mathcal{P}_{Y}^{n} \succeq V) = P (y(x) \succeq V, V \in \mathcal{R}_{n}).
\]

The value of \( I_{WH}(y) \) with \( \zeta = 1 \) for \( y \in [0, 1]^{2} \) with a four-point current Pareto front is plotted in Figure 2.6.

What does the Gaussian weighted hypervolume improvement offer that the hypervolume improvement does not? The main advantage is that \( I_{WH}(\cdot) \) is a positive
Figure 2.6: Surface plot of $I_{WH}(\mathbf{y})$ with $\zeta = 1$ when $m = 2$. The black squares represent the current Pareto front.

function for all $\mathbf{y}(\mathbf{x}) \in \mathcal{R}_n$, while $I_{WH}(\cdot)$ can be zero for $\mathbf{y}(\mathbf{x}) \in \mathcal{R}_n$ that do not dominate the chosen reference point $R$. For example, in Figure 2.2, the output vector $\mathbf{y} = (1,0)$ yields $I_{WH}(\mathbf{y}) = 0$, despite being in the nondominated region, because it does not dominate $(0.9,0.9)$. It seems unsatisfactory that a nondominated output yields exactly the same improvement as a dominated output. The same point $\mathbf{y}$, however, will always yield positive, albeit possibly very small, improvement when using $I_{WH}(\cdot)$. 
Implementation - QIM1

In this case, \( QI(x) \) is defined as

\[
E \left[ I_{WH} \left( Y(x) \right) \mid Y^{m,n} = y^{m,n} \right]. \tag{2.3.10}
\]

Changing the order of integration and letting \( f(y(x)\mid y^{m,n}) \) be the normal density function of \( Y(x) \) given \( y^{m,n} \) specified in (1.3.16)-(1.3.17), we can express (2.3.10) as

\[
E \left[ I_{WH} \left( Y(x) \right) \mid Y^{m,n} = y^{m,n} \right] = E \left[ P \left( Y(x) \geq V, V \in \mathbb{R}_n \right) \mid Y^{m,n} = y^{m,n} \right]
\]

\[
= \int_{\mathbb{R}_n} \int_{\mathbb{R}_m} 1_{[y(x) \geq v]} \prod_{i=1}^{m} 1_{\xi \phi \left( \frac{y_i}{\zeta} \right)} f(y(x)\mid y^{m,n}) \, dv \, dy(x)
\]

\[
= \int_{\mathbb{R}_m} 1_{v \in \mathbb{R}_n} \left( \int_{\mathbb{R}_m} 1_{[y(x) \geq v]} \prod_{i=1}^{m} 1_{\xi \phi \left( \frac{y_i}{\zeta} \right)} f(y(x)\mid y^{m,n}) \, dy(x) \right) \prod_{i=1}^{m} 1_{\xi \phi \left( \frac{y_i}{\zeta} \right)} \, dv
\]

\[
= \int_{\mathbb{R}_m} 1_{v \in \mathbb{R}_n} P \left( Y_1(x) \leq v_1, \ldots, Y_m(x) \leq v_m \mid Y^{m,n} = y^{m,n} \right) \, dv \tag{2.3.11}
\]

One strategy for obtaining a sample average approximation to (2.3.11) is as follows. Generate an independent, identically distributed sample \( V^1, \ldots, V^T \) from a \( N(0_m, \zeta^2 I_m) \) distribution. Then, approximate (2.3.10) with

\[
\frac{1}{T} \sum_{i=1}^{T} 1_{v \in \mathbb{R}_n} P \left( Y_1(x) \leq v_1, \ldots, Y_m(x) \leq v_m \mid Y^{m,n} = y^{m,n} \right), \tag{2.3.12}
\]

where \( P \left( Y_1(x) \leq v_1, \ldots, Y_m(x) \leq v_m \mid Y^{m,n} = y^{m,n} \right) \) can be calculated numerically (using Matlab’s \texttt{mvncdf} function, for example). However, since the indicator function \( 1_{v \in \mathbb{R}_n} \) does not depend on \( x \), we can speed up calculations by checking whether or not each \( V^i \) is dominated before calling our direct search optimization algorithm. Among the \( T \) output samples \( V^1, \ldots, V^T \), remove all vectors such that \( 1_{v \in \mathbb{R}_n} = 0 \). Relabel this smaller sample as

\[ V^{k_1}, \ldots, V^{k_N} \]
(N \leq T). Now, approximate (2.3.10) by
\[ \frac{1}{T} \sum_{i=1}^{N} P \left( Y_i(x) \leq V^{k_i}_1, \ldots, Y_m(x) \leq V^{k_i}_m | Y^{m,n} = y^{m,n} \right). \] (2.3.13)
instead of (2.3.12), and apply the direct search optimization algorithm to (2.3.13).
Moreover, if we make the simplifying assumption that the m outputs are independent, then we can express the normal probability as a product of univariate normal cdf’s, and the sample average approximation (2.3.13) becomes
\[ \frac{1}{T} \sum_{i=1}^{N} \prod_{j=1}^{m} \Phi \left( \frac{V^{k_i}_j - \hat{y}_j(x)}{s_i(x)} \right), \] (2.3.14)
where
\[ \hat{y}_j(x) = E \left[ Y(x) | Y^{m,n} = y^{m,n} \right] \]
\[ s^2_i(x) = \text{Cov} \left( Y(x) | Y^{m,n} = y^{m,n} \right). \]
The expression (2.3.14) is faster to evaluate than (2.3.13) because the product univariate normal cdf’s can be calculated much more quickly than multivariate normal probabilities.

**Implementation - QIM2**

In this case, \( QI(x) \) is defined as
\[ QI(x) = P \left( Y(x) \in \mathcal{R}_n | Y^{m,n} = y^{m,n} \right) I_{\mathcal{W}H}(\overline{Y}(x)). \]
Even though \( QI(x) \) is not an expectation in the QIM2 setting, we can still set construct an approximation to \( QI(x) \) where all random samples are drawn before running the deterministic direct search optimization algorithm. To do so, first construct the approximations for \( P \left( Y(x) \in \mathcal{R}_n | Y^{m,n} = y^{m,n} \right) \) and \( \overline{Y}(x) \) described in (2.2.4)-(2.2.7). Then, generate an independent, identically distributed sample \( V^1, \ldots, V^T \)
from a \( N (0_m, \zeta^2 I_m) \) distribution. Since \( \overline{Y}(x) \in \mathcal{R}_n \), we can approximate \( I_{WH} (\overline{Y}(x)) \) by

\[
\tilde{I}_{WH} (\overline{Y}(x)) = \frac{1}{T} \sum_{t=1}^{T} 1_{\left[ \overline{y}(x) \succ V_t \right]}.
\]

We then approximate \( QI(x) \) by

\[
\tilde{QI}(x) = \hat{P} (Y(x) \in \mathcal{R}_n | Y^{m,n} = y^{m,n}) \tilde{I}_{WH} (\overline{Y}(x)).
\]

Just as for QIM1, \( \tilde{QI}(x) \) is a deterministic function for a particular realization any initially drawn random samples.

### 2.3.3 Completeness Indicator Improvement

With the exception of the 0-1 improvement function \( I_P(\cdot) \) used to define the probability of improvement, all of the improvement functions presented thus far depend on the relative scaling of the various outputs. In this section, we will introduce a scaling-invariant alternative that is based on the completeness indicator (1.5.2). Unlike \( I_P(\cdot) \), the improvement functions presented here are able to quantify the magnitude of improvement, and not merely whether or not improvement occurs.

First, let \( U \) be a uniformly distributed random vector on the input space \( \mathcal{X} \) independent of the process \( Y(\cdot) \). We define the completeness indicator improvement as

\[
I_C (y(x)) = I_{CP} (\mathcal{P}_{\mathcal{Y}}^{n+1}(y(x))) - I_{CP} (\mathcal{P}_{\mathcal{Y}}^{n}).
\]  

Using Theorem 6 with \( y(U) \) playing the role of \( W \), we can show that

\[
I_C (y(x)) = P (\mathcal{P}_{\mathcal{Y}}^{n+1}(y(x)) \succeq y(U)) - P (\mathcal{P}_{\mathcal{Y}}^{n} \succeq y(U)) = P (y(x) \succeq y(U), y(U) \in \mathcal{R}_n).
\]
Unlike the other improvement functions, which in principal only require knowledge of \( y(x) \) and \( P^n \) to compute, (2.3.16) requires complete knowledge of the input-output relationship between \( x \) and \( y(\cdot) \). Therefore, any interpretation of the expected improvement at \( x \) must not only account for the uncertainty in \( y(\cdot) \) at \( x \), but also for the uncertainty at any element of \( X \). We propose two ways to do this. First, we introduce the \textit{averaged completeness indicator}, which accounts for the uncertainty in the input-output relationship by taking the expectation of (2.3.16) with respect to the entire posterior predictive process. Secondly, we introduce the \textit{estimated completeness indicator}, which \textit{plugs-in} the posterior predictive mean \( \hat{y}(U) \) for \( y(U) \) in (2.3.16). In the following, we will describe how to implement these approaches in both the QIM1 and QIM2 framework.

**Implementation - QIM1 (Averaged Completeness Indicator)**

Here, \( QI(x) \) is defined as \( E^P [I_C(Y(x))] \) where \( E^P \) represents expectation with respect to the posterior predictive process \( [Y(\cdot)|Y_{m,n} = y_{m,n}] \). Thus, we account for the uncertainty in \( Y(\cdot) \) at any element of \( X \) by averaging \( I_C(Y(x)) \) with respect to the posterior predictive process. Letting

\[
|X| = \int_X du
\]

and changing the order of integration, we have the following equivalent expression:

\[
E^P[I_C(Y(x))]
\]

\[
= E^P \left[ E^U \left[ 1_{[Y(x) \geq Y(U)], Y(U) \in \mathbb{R}^n]} \right] \right]
\]

\[
= E^P \left[ \int_X 1_{[Y(x) \geq Y(u), Y(u) \in \mathbb{R}^n]} \frac{1}{|X|} du \right]
\]

\[
= \int_X E^P \left[ 1_{[Y(x) \geq Y(u)], Y(u) \in \mathbb{R}^n]} \frac{1}{|X|} du \right]
\]

84
\[
= \int_{\mathcal{X}} E^P \left[ E^P \left[ 1_{\{Y(x) \geq Y(u)\}} 1_{\{Y(u) \in \mathbb{R}^n\}} Y(u) \right] \right] \frac{1}{|\mathcal{X}|} du \\
= \int_{\mathcal{X}} E^P \left[ 1_{\{Y(u) \in \mathbb{R}^n\}} E^P \left[ 1_{\{Y(x) \geq Y(u)\}} Y(u) \right] \right] \frac{1}{|\mathcal{X}|} du \\
= \int_{\mathcal{X}} E^P \left[ 1_{\{Y(u) \in \mathbb{R}^n\}} P\left( Y_1(x) \leq Y_1(u), \ldots, Y_m(x) \leq Y_m(u) | Y(u), Y^{m,n} \right) \right] \frac{1}{|\mathcal{X}|} du.
\]

One strategy for obtaining a sample average approximation to \( E^P \left[ I_C(Y(x)) \right] \) is the following. Generate an independent, identically distributed sample \( U^1, \ldots, U^T \) from a uniform distribution on \( \mathcal{X} \). Then, for each \( U^i \), generate an independent, identically distributed sample \( Z^1_i, \ldots, Z^S_i \) from a \( N(0_m, I_m) \) distribution. Given \( U^i \) and letting \( C(U^i) \) be the Cholesky decomposition of \( S(U^i) \), transform \( Z^j_i \) into a random variable \( Y^j_i(U^i) = C(U^i)Z^j_i + \hat{y}(U^i) \sim N(\hat{y}(U^i), S(U^i)) \). Using multivariate normal theory and the notation introduced in (1.3.11) and (1.3.12), we have

\[
\left[ Y(x) | Y^{m,n}, Y^j(U^i) \right] \sim N(\mu_{x|U^i}, \Sigma_{x|U^i})
\] (2.3.17)

where

\[
\mu_{x|U^i} = \hat{y}(x) + S(x, U^i)S^{-1}(U^i) \left( Y^j(U^i) - \hat{y}(U^i) \right)
\] (2.3.18)

and

\[
\Sigma_{x|U^i} = S(x) - S(x, U^i)S^{-1}(U^i)S^T(x, U^i).
\] (2.3.19)

Therefore, \( P \left( Y_1(x) \leq Y^j_1(U^i), \ldots, Y_m(x) \leq Y^j_m(U^i) | Y^{m,n}, Y^j(U^i) \right) \) can be approximated numerically, using Matlab’s \text{mvncdf} \ function, for example. Then, we can use

\[
\frac{1}{ST} \sum_{i=1}^T \sum_{j=1}^S P \left( Y_1(x) \leq Y^j_1(U^i), \ldots, Y_m(x) \leq Y^j_m(U^i) | Y^{m,n}, Y^j(U^i) \right) 1_{[Y^j(U^i) \in \mathbb{R}^n]}
\] (2.3.20)
as our sample average approximation to $E^P [I_C (Y(x))]$.

We can employ a much more efficient version of (2.3.20) that exploits the fact that $1_{Y^j(U) \in \mathcal{R}_n}$ does not depend on $x$. Prior to starting the direct search optimization algorithm, among the $ST$ input-output samples $(U^i, Y^j(U^i))$, remove all elements for which $1_{Y^j(U^i) \in \mathcal{R}_n} = 0$. Relabel this smaller sample as

$$(U^{k_1}, Y^{k_1}(U^{k_1})), \ldots, (U^{k_N}, Y^{k_N}(U^{k_N}))$$

($N \leq ST$). Now, apply the direct search algorithm to the approximation

$$\frac{1}{ST} \sum_{i=1}^{N} P \left( Y_1(x) \leq Y^{k_i}_1(U^{k_i}), \ldots, Y_m(x) \leq Y^{k_i}_m(U^{k_i}) | Y^{m,n}, Y^{k_i}(U^{k_i}) \right). \quad (2.3.21)$$

Notice that all of the checks for nondomination are performed outside the direct search optimization algorithm.

If we assume that $Y_1(\cdot) \ldots Y_m(\cdot)$ are independent GaSPs with process variances $\sigma_1^2, \ldots, \sigma_m^2$ and correlation functions $R_1(\cdot), \ldots, R_m(\cdot)$, we can obtain an even simpler sample average approximation for $E^P [I_C (Y(x))]$. First, notice that for any $u$ and $x$ in $\mathcal{X}$, we have

$$[Y_i(x), Y_i(u) | Y^{m,n}] \sim N \left( \left[ \begin{array}{c} \hat{y}_i(x) \\ \hat{y}_i(u) \end{array} \right], \sigma_i^2 \left[ \begin{array}{cc} r_i(x,x) & r_i(x,u) \\ r_i(x,u) & r_i(u,u) \end{array} \right] \right), \quad (2.3.22)$$

(see section 1.3.1), and $[Y_i(x), Y_i(u) | Y^{m,n}] \perp [Y_j(x), Y_j(u) | Y^{m,n}]$ for $i \neq j$. Again, generate an independent, identically distributed sample $Z^1, \ldots, Z^S$ from a $N(0_m, I_m)$ distribution, and also generate an independent, identically distributed sample $U^1, \ldots, U^T$ from a uniform distribution on $\mathcal{X}$. For any given $U^i$, letting $s_j(U^i) = \sqrt{\sigma_i^2 r_i(U^i)}$ be the standard deviation of $Y_j(U^i)$ conditional on $Y^{m,n}$, we can transform each $Z_j^k$ (where $Z_j^k$ is the $j^{th}$ component of $Z^k$) into a random variable $Y_j^k(U^i) = \hat{y}_j(U^i) + s_j(U^i) Z_j^k \sim N(\hat{y}_j(U^i), s_j^2(U^i))$. Therefore, for each $j = 1, \ldots, m$, and $k = 1, \ldots, T$
we have \( [Y_j(x)|Y_j^k(U^i), Y^{m,n}] \sim N \left( \mu_{x|U^i}, \left( \sigma_{x|U^i} \right)^2 \right) \), where
\[
\mu_{x|U^i}^j = \hat{y}_i(x) + \frac{r_i(x,u)}{r_i(u)} (Y_j^k(U^i) - \hat{y}_i(u))
\]
and
\[
\left( \sigma_{x|U^i}^j \right)^2 = \sigma_i^2 \left( \frac{r_i(x,u)}{r_i(u)} \right).
\]
Because of the independence of the \( m \) processes, we have
\[
P \left( Y_1(x) \leq Y_1^j(U^i), \ldots, Y_m(x) \leq Y_m^j(U^i) | Y^{m,n}, Y^j(U^i) \right) = \prod_{k=1}^m \Phi \left( \frac{Y_k^j(U^i) - \mu_{x|U^i}^k}{\sigma_{x|U^i}^k} \right).
\]
In this case, instead of having to compute a multivariate cdf as in (2.3.20), we can compute a product of univariate normal cdfs. We can then use
\[
\frac{1}{ST} \sum_{i=1}^T \sum_{j=1}^S \prod_{k=1}^m \Phi \left( \frac{Y_k^j(U^i) - \mu_{x|U^i}^k}{\sigma_{x|U^i}^k} \right) 1[Y^j(U^i) \in \mathbb{R}_n] \tag{2.3.23}
\]
as our sample average approximation to \( E^P [I_c (Y(x))] \). Just as in the dependence case, we must evaluate (2.3.23) many times in our direct search optimization algorithm for different values of \( x \). To speed computation, we can delete all input-output draws \( (U^i, Y^j(U^i)) \) such that \( 1[Y^j(U^i) \in \mathbb{R}_n] = 0 \) before starting our optimization algorithm.

Doing so, and relabeling the remaining nondominated input-output samples as
\[
(U^{k_1}, Y_1^{k_1} (U^{k_1}) \ldots Y_m^{k_1} (U^{k_1})), \ldots, (U^{k_N}, Y_1^{k_N} (U^{k_N}) \ldots Y_m^{k_N} (U^{k_N})) ,
\]
we then can apply the direct search algorithm to
\[
\frac{1}{ST} \sum_{i=1}^N \prod_{j=1}^m \Phi \left( \frac{Y_j^k(U^{k_i}) - \mu_{x|U^{k_i}}^j}{\sigma_{x|U^{k_i}}^j} \right). \tag{2.3.24}
\]
Implementation - QIM2 (Averaged Completeness Indicator)

In the QIM2 framework described in Section 2.1, $QI(x)$ with the completeness indicator improvement function would be defined as

$$QI(x) = P(\mathbf{Y}(x) \in \mathcal{R}_n | \mathbf{Y}^{m,n} = \mathbf{y}^{m,n}) I_C(\overline{\mathbf{Y}}(x)). \tag{2.3.25}$$

Unfortunately, $I_C(\overline{\mathbf{Y}}(x))$ cannot be computed unless we have complete understanding of the input-output relationship between $x$ and $y(\cdot)$. Here, we deal with this uncertainty by taking the expectation of $I_C(\overline{\mathbf{Y}}(x))$ with respect to the distribution of $[\mathbf{Y}(\cdot)|\mathbf{Y}^{m,n} = \mathbf{y}^{m,n}]$. So, just as in the QIM1 implementation, we account for uncertainty in $\mathbf{Y}(\cdot)$ by averaging $I_C(\overline{\mathbf{Y}}(x))$ with respect to the posterior predictive distribution. We then have

$$QI(x) = P(\mathbf{Y}(x) \in \mathcal{R}_n | \mathbf{Y}^{m,n} = \mathbf{y}^{m,n}) E^P[I_C(\overline{\mathbf{Y}}(x))],$$

$$= P(\mathbf{Y}(x) \in \mathcal{R}_n | \mathbf{Y}^{m,n} = \mathbf{y}^{m,n})$$

$$\times E^P\left[ \int_{\mathcal{X}} 1[\overline{\mathbf{Y}}(x) \geq \mathbf{Y}(u)] 1[\mathbf{Y}(u) \in \mathcal{R}_n] \frac{1}{|\mathcal{X}|} d\mathbf{u} \right],$$

$$= P(\mathbf{Y}(x) \in \mathcal{R}_n | \mathbf{Y}^{m,n} = \mathbf{y}^{m,n})$$

$$\times \int_{\mathcal{X}} E^P\left[ 1[\overline{\mathbf{Y}}(x) \geq \mathbf{Y}(u)] 1[\mathbf{Y}(u) \in \mathcal{R}_n] \right] \frac{1}{|\mathcal{X}|} d\mathbf{u},$$

where $E^P$ represents expectation with respective to the posterior predictive process $[\mathbf{Y}(\cdot)|\mathbf{Y}^{m,n} = \mathbf{y}^{m,n}]$.

Even though $QI(x)$ is not an expectation, we can still set up an approximation that is similar in spirit to a sample average approximation. First, generate independent random samples

$$U^1, \ldots, U^T \overset{iid}{\sim} \text{Uniform}(\mathcal{X})$$

$$Z_{x1}^1, \ldots, Z_{xV}^V \overset{iid}{\sim} N(\mathbf{0}_m, \mathbf{I}_m),$$

88
Then, for any \( x \), letting \( C(x) \) be the Cholesky decomposition of \( S(x) \), we can transform each \( Z^i_x \) into a random variable \( Y^j(x) = C(x)Z^i_x + \hat{y}(x) \sim N(\hat{y}(x), S(x)) \). So, the \( Y^j(x) \)'s are draws from the posterior predictive distribution at \( x \). We can use these samples to obtain the following approximations to \( P(Y(x) \in R_n | Y^{m,n} = y^{m,n}) \) and \( \bar{Y}(x) \):

\[
\hat{P}(Y(x) \in R_n | Y^{m,n} = y^{m,n}) = \frac{1}{V} \sum_{j=1}^{V} 1[Y^j(x) \in R_n] \quad \text{and} \quad \bar{Y}(x) = \frac{\sum_{j=1}^{V} Y^j(x) 1[Y^j(x) \in R_n]}{\sum_{j=1}^{V} 1[Y^j(x) \in R_n]}
\]

Next, for each \( U^i \), generate an independent, identically distributed sample \( Z^i_{1}, \ldots, Z^i_S \overset{iid}{\sim} N(0_m, I_m) \).

Given \( U^i \), let \( C(U^i) \) be the Cholesky decomposition of \( S(U^i) \). Transform each \( Z^i_j \) into a random variable \( Y^j(U^i) = C(U^i)Z^i_j + \hat{y}(U^i) \sim N(\hat{y}(U^i), S(U^i)) \). We can then approximate \( QI(x) \) as

\[
\hat{QI}(x) = \hat{P}(Y(x) \in R_n | Y^{m,n} = y^{m,n}) \frac{1}{ST} \sum_{i=1}^{T} \sum_{j=1}^{S} 1[\bar{Y}(x) \geq Y^j(U^i)] 1[Y^j(U^i) \in R_n].
\]

As with the QIM1 implementation, delete all input-output draws \( (U^i, Y^j(U^i)) \) for which \( 1[Y^j(U^i) \in R_n] = 0 \) before starting our optimization algorithm to speed computations. Doing so, and relabeling the remaining nondominated input-output samples as

\[
(U^{k_1}, Y^{k_1}(U^{k_1})), \ldots, (U^{k_N}, Y^{k_N}(U^{k_N}))
\]

(where \( N \leq ST \)), we then can apply the direct search algorithm to maximize the approximation

\[
\hat{QI}(x) = \hat{P}(Y(x) \in R_n | Y^{m,n} = y^{m,n}) \frac{1}{ST} \sum_{i=1}^{N} 1[\bar{Y}(x) \geq Y^{k_i}(U^{k_i})].
\]
Implementation - QIM1 (Estimated Completeness Indicator)

In this case, we first define the estimated completeness indicator as

$$
\hat{I}_C(y(x)) = P(y(x) \geq \hat{y}(U), \hat{y}(U) \in \mathcal{R}_n).
$$

(2.3.26)

Then, $QI(x)$ is defined as

$$
QI(x) = E \left[ \hat{I}_C(Y(x)) \middle| Y_{m,n} = y^{m,n} \right].
$$

(2.3.27)

Therefore, we deal with the fact that $y(U)$ is unknown by plugging-in the posterior predictive mean

$$
\hat{y}(U) = E^P[Y(U)],
$$

where $E^P$ represents expectation with respect to the posterior predictive process $[Y(\cdot)|Y_{m,n} = y^{m,n}]$.

Here is a strategy for obtaining a sample average approximation to (2.3.27). We first generate an independent, identically distributed sample $U^1, \ldots, U^T$ from a uniform distribution over $X$. Then, we calculate the posterior mean of each sample, and denote these values as $\hat{y}(U^1), \ldots, \hat{y}(U^T)$.

For any given $x$, we can construct a sample $Y^1(x), \ldots, Y^S(x)$ from the posterior predictive distribution of $Y(x)$ from an initial random sample $Z^1, \ldots, Z^S$, as described in (2.2.2) - (2.2.3). For any $Y^s(x)$, $\hat{I}_C(y(x))$ can be approximated by

$$
\hat{I}_C(Y^s(x)) = \frac{1}{T} \sum_{t=1}^{T} 1[y^s(x) \geq \hat{y}(U^t)] 1[\hat{y}(U^t) \in \mathcal{R}_n]
$$

We can then use

$$
\overline{QI(x)} = \frac{1}{S} \sum_{s=1}^{S} \hat{I}_C(Y^s(x))
$$

$$
= \frac{1}{ST} \sum_{s=1}^{S} \sum_{t=1}^{T} 1[y^s(x) \geq \hat{y}(U^t)] 1[\hat{y}(U^t) \in \mathcal{R}_n]
$$

(2.3.28)
as our sample average approximation to (2.3.27).

Just as we did with the averaged completeness indicator, we can employ a much more efficient version of (2.3.28) that exploits the fact that \(1\mathbb{1}_{\hat{g}(U^t) \in \mathbb{R}_n}\) does not depend on \(x\). Prior to starting the direct search optimization algorithm, among the \(T\) samples \(\hat{g}(U^1), \ldots, \hat{g}(U^T)\), remove all elements such that \(1\mathbb{1}_{\hat{Y}(U^t) \in \mathbb{R}_n} = 0\). Relabel this smaller sample as \(\hat{g}(U^{k_1}), \ldots, \hat{g}(U^{k_N})\) \((N \leq T)\). Now, apply the direct search algorithm to

\[
\frac{1}{ST} \sum_{s=1}^{S} \sum_{t=1}^{N} 1\mathbb{1}_{\hat{y}^s(x) \geq \hat{g}(U^{k_t})}. \tag{2.3.29}
\]

Notice that all of the checks for nondomination are performed outside the direct search optimization algorithm.

**Implementation - QIM2 (Estimated Completeness Indicator)**

In the QIM2 framework described in Section 2.1, \(QI(x)\) is defined as

\[
QI(x) = P(Y(x) \in \mathbb{R}_n | Y^{m,n} = y^{m,n}) \tilde{I}_C(\overline{Y}(x)). \tag{2.3.30}
\]

Even though \(QI(x)\) is not an expectation in the QIM2 setting, we can still set up an approximation to \(QI(x)\) where all random sample are drawn before running the deterministic direct search optimization algorithm. To do so, first construct approximation for \(P(Y(x) \in \mathbb{R}_n | Y^{m,n} = y^{m,n})\) and \(\overline{Y}(x)\), as described in (2.2.4)-(2.2.7). Then, generate an independent, identically distributed sample \(U^1, \ldots, U^T\) from a from a uniform distribution over \(\mathcal{X}\). We calculate the posterior mean of each sample, and denote these values as

\[
\hat{g}(U^1), \ldots, \hat{g}(U^T). \tag{2.3.31}
\]
Removing the dominated vectors from (2.3.31) and relabeling the non-dominated vectors as

\[ \hat{y}(U^{k_1}), \ldots, \hat{y}(U^{k_N}) \]

(where \( N \leq T \)) as in the previously described completeness indicator implementations, we can approximate \( QI(x) \) with

\[
\hat{QI}(x) = \hat{P}(Y(x) \in \mathcal{R}_n | Y_{m,n} = y_{m,n}) \times \frac{1}{N} \sum_{t=1}^{N} \mathbb{1}_{[\hat{Y}(x) \succeq \hat{y}(U^{k_t})]},
\]

(2.3.32)

which is a deterministic function for a particular realization of \( U^1, \ldots, U^T \) and \( Z^1, \ldots, Z^S \).

### 2.4 Comparison of Improvement Functions

Thus far, we have presented four improvement functions from the literature (\( I_{PI}(\cdot) \), \( I_{K}(\cdot) \), \( I_{H}(\cdot) \), and \( I_{P}(\cdot) \)) and introduced three new ones (\( I_{M}(\cdot) \), \( I_{WH}(\cdot) \), and \( I_{C}(\cdot) \)).

There are two ways to judge the quality of the various improvement functions. One approach is first to define some theoretical properties that would make an attractive improvement function, and then determine which improvement functions possess these desirable properties. This is the focus of the current section. The other way to judge their quality is by using the various improvement functions in expected improvement algorithms to approximate the Pareto front of a set of real-world multiobjective optimization problems and a test bed of problems from the literature, and then compare the various Pareto front approximations via Pareto set quality indicators and plots. This is the focus of Chapter 3.
2.4.1 Desirable Properties for Improvement Functions

1. **Monotonicity**: An improvement function $I^*(\cdot)$ should be *monotonic with respect to Pareto dominance*. That is, for any $y(x)$ and $y(x')$, $y(x) \succeq y(x')$ implies $I^*(y(x)) \geq I^*(y(x'))$. This criterion says that for any two objective space vectors, if one weakly dominates the other, than the dominated vector should not have greater improvement.

2. **Generalization of $I(x)$**: The multiobjective expected improvement algorithm and associated improvement function should reduce to the Jones-Schonlau single-objective expected improvement algorithm and associated improvement function employed in Section 1.4 when $m = 1$. The rationale behind this criterion is that if a proven optimization algorithm for computer experiments with a single output was just a special case (with $m = 1$) of a proposed multiobjective optimization algorithm, one would be able to make the argument that the proposed multiobjective expected improvement algorithm is indeed a reasonable generalization.

3. **Scale Invariance**: The improvement function and associated expected improvement algorithm should not depend on the scaling of the output, as we may have little prior information regarding the scales of the various outputs. The discussion in the Nowacki beam example of Section 3.2.2 illustrates why this criterion is relevant by showing that two seemingly reasonable output-scaling strategies can lead to very different Pareto front approximations.

4. **Free of External Parameters**: The improvement function should not depend on any user-supplied parameters, such as a bounding reference point. The
discussion in the Nowacki beam example of Section 3.2.2 illustrates why this
criterion is relevant by showing that two seemingly reasonable choices of refer-
ence points for the hypervolume improvement function can lead to very different
Pareto front approximations.

5. **Relationship to a Pareto Set Quality Indicator**: Since the main goal is
to approximate the Pareto set (and the corresponding Pareto front), the im-
provement functions should seek to maximize some a measure of the increase in
Pareto set quality between the current Pareto front and the Pareto front after
augmenting the current design by \( y(x) \). For example, the approach taken by
Emmerich et al. (2006) explicitly defines improvement at \( y(x) \) as the increase
in the hypervolume indicator (a popular Pareto set approximation quality indi-
cator) if \( P \) is augmented by \( y(x) \).

6. **Exact Computability**: Ideally, we would like to be able to compute the de-
sired improvement function directly, without having to rely on an approxima-
tion.

### 2.4.2 Properties of Improvement Functions

**\( I_{P_T}(\cdot) \): 0-1 Improvement**

1. **Monotonicity**: Trivially, \( I_{P_T}(\cdot) \) is monotonic with respect to Pareto dom-
nance. To see this, notice that if \( y(x) \succeq y(x') \), then \( I_{P_T}(x) = I_{P_T}(x') \) if both
outputs are in \( R_n \) or both outputs are \( R_n^c \). If \( y(x) \in R_n \) and \( y(x') \notin R_n \), then

\[
I_{P_T}(x) = 1 > 0 = I_{P_T}(x').
\]
2. **Generalization of** $I(x)$: When $m = 1$, $1_{P_I(y(x) = 1_{|y(x) < y_{min}})}$ and

$P(Y(x) \in \mathcal{R}_n|Y^{m,n} = y^{m,n})$ is equivalent to the single-objective probability of improvement rather than the expected improvement. Forrester et al. (2008) claims that the probability of improvement is less effective as a sequential design strategy for single-objective optimization because it only takes into account whether or not an improvement occurs and ignores the magnitude of the improvement.

3. **Scale Invariance**: $I_{P_I}(\cdot)$ and $P(Y(x) \in \mathcal{R}_n|Y^{m,n} = y^{m,n})$ do not depend on the scaling of the output, which is the main advantage of the probability of improvement according to Keane (2006).

4. **Free of External Parameters**: $I_{P_I}(\cdot)$ does not depend on any user-supplied external parameters.

5. **Relationship to a Pareto Set Quality Indicator**: It does not appear that Keane (2006) constructed this improvement function with any Pareto set approximation quality indicator in mind, and there does not appear to be a meaningful connection to one.

6. **Exact Computability**: An advantage of $I_{P_I}(\cdot)$ is that it is quick and straightforward to compute exactly. All that is required is a check that $y$ is not dominated by an vector in $P^n_y$. 

95
Figure 2.7: A \preceq B but $I_K(A) > I_K(B)$

$I_K(\cdot)$: Keane’s Distance Based Improvement

1. **Monotonicity**: $I_K(\cdot)$ is *not* monotonic with respect to Pareto dominance. Consider the following counterexample with two dimensional output, which is displayed in Figure 2.7. Assume that the current Pareto front is

$$\{(0.5, 0.8), (0.55, 0.75), (0.65, 0.65), (0.75, 0.55)\}.$$  

We want to evaluate the improvement upon this current Pareto front at $A = (0.45, 1)$ and $B = (0.35, 0.75)$ respectively. One can easily see that $A \preceq B$. But, $I_K(A) > I_K(B)$, as $A$ is further than $B$ from $(0.5, 0.8)$, the point on the current Pareto front that is closest to both $A$ and $B$.  

96
2. Generalization of $I(x)$: $I_K(y(x)) = I(x)$ when $m = 1$, as

$$
\min_{x_i \in P_x} \sqrt{\sum_{k=1}^{m} (y_k(x^*) - y_k(x_i))^2} = \sqrt{(y(x^*) - y_{min}^n)^2} = y_{min}^n - y(x^*)
$$

if $y_{min}^n - y(x^*) > 0$. This means that in the single-objective case,

$$
I_K(y) = \begin{cases} 
  y_{min}^n - y & \text{if } y_{min}^n - y > 0 \\
  0 & \text{otherwise,}
\end{cases}
$$

which is equivalent to $I(x)$. Therefore, the improvement function $I(x)$ is just a special case of $I_K(\cdot)$.

3. Scale Invariance: Keane (2006) points out that his QIM2 implementation $QI(x)$ is dependent on the scaling of the output. In general, $I_K(\cdot)$ is scaling dependent.


5. Relationship to a Pareto Set Quality Indicator: It does not appear that Keane (2006) constructed this improvement function with any Pareto set approximation quality indicator in mind, and there does not appear to be a meaningful connection to one.

6. Exact Computability: $I_K(y(x))$ is straightforward to compute, and it is relatively fast to compute, since it only requires computing the distance from $y(x)$ to all points in $P^*_y$ and then choosing the smallest distance. No approximations are required.
$I_H(\cdot)$: Hypervolume Improvement

1. **Monotonicity**: Recall from Chapter 1 that, for two Pareto front approximation $A$ and $B$, if $A \succ B$ then $I_H(A) > I_H(B)$. This means that if $y(x) \succeq y(x')$ and both are in the nondominated region $R_n$ and both dominate $R$, then

$$I_H(y(x)) - I_H(y(x')) = I_H(\{y(x)\} \cup P_n^y, R) - I_H(P_n^y, R)$$

$$- (I_H(\{y(x')\} \cup P_n^y, R) - I_H(P_n^y, R))$$

$$= I_H(\{y(x)\} \cup P_n^y, R) - I_H(\{y(x')\} \cup P_n^y, R)$$

$$\geq 0$$

since

$$\{y(x')\} \cup P_n^y \succeq \{y(x)\} \cup P_n^y.$$  

The above inequality holds trivially if $y(x)$ and $y(x')$ are in other regions of the objective space as well.

2. **Generalization of $I(x)$**: When $m = 1$,

$$I_H(y(x))$$

$$= (R - \min(y(x), y_{min})) - (R - y_{min})$$

$$= \begin{cases} 
    y_{min} - y(x) & \text{if } y_{min} - y(x) > 0 \\
    0 & \text{otherwise}
\end{cases}$$

So, $I_H(y(x))$ reduces to $I(x)$ when $m = 1$.

3. **Scale Invariance**: A weakness of the hypervolume improvement function is that the hypervolume indicator is scaling dependent. Emmerich et al. (2006) does not detail how the scaling issues are resolved.
4. **Free of External Parameters**: The hypervolume improvement function requires a user-supplied reference set that bounds the objective space.

5. **Relationship to a Pareto Set Quality Indicator**: The main strength of the hypervolume improvement function is that it explicitly employs the only unary Pareto set approximation quality indicator that is strictly monotonic. Another nice property, pointed out in Emmerich et al. (2006), is that if the objective space is countable, the hypervolume indicator is maximized at the true Pareto front. Aside from its theoretical properties, the hypervolume indicator is one of the most popular quality indicators in the multiobjective evolutionary algorithm community according to Bader and Zitzler (2009).

6. **Exact Computability**: A major weakness of the hypervolume improvement function is its computational complexity. According to Fonseca et al. (2006), the best algorithms for calculating the hypervolume indicator can only be solved in $O\left(p^{m/2}\log(p)\right)$-time, where $p$ is the size of $\mathcal{P}_y$. For even moderate sized objective spaces, one usually must resort estimating the hypervolume indicator via Monte Carlo methods. The computational demands associated with hypervolume computations make the QIM2 definition of $QI(x)$ particularly attractive.

$I_P(\cdot)$: Pareto Improvement Function

1. **Monotonicity**: $I_P(\cdot)$ is monotonic with respect to Pareto dominance. To prove this, let $y(x)$ and $y(x')$ be two objective space vectors such that $y(x) \preceq y(x')$. We have

$$I_P(y(x)) = -\max_{x_i \in \mathcal{P}_x} \min_{j=1,\ldots,m} (y_j(x) - y_j(x_i))$$
\[
\begin{align*}
&= \min_{x_i \in \mathcal{P}^n} \max_{j=1,\ldots,m} (y_j(x_i) - y_j(x)) \\
&\leq \min_{x_i \in \mathcal{P}^n} \max_{j=1,\ldots,m} (y_j(x_i) - y_j(x')) \\
&= -\max_{x_i \in \mathcal{P}^n} \min_{j=1,\ldots,m} (y_j(x') - y_j(x_i)) \\
&= I_P(y(x')).
\end{align*}
\]

Therefore, \(I_P(\cdot)\) is monotonic with respect to Pareto dominance.

2. Generalization of \(I(x)\): When \((m = 1)\), notice that

\[
I_P(x) = -\max_{x_i \in \mathcal{P}^n} (y(x) - y(x_i)) \\
= \min_{x_i \in \mathcal{P}^n} (y(x_i) - y(x)) \\
= y_{\text{min}}^n - y(x) \\
\neq y_{\text{min}}^n - y(x)I[y_{\text{min}}^n - y(x)<0] \\
= I(y(x))
\]

for any \(x\) such that \(y_{\text{min}}^n - y(x) < 0\). Therefore, \(I_P(\cdot)\) is not equivalent to (1.4.7) in the single-objective case because it yields negative values at dominated solutions instead of assigning them a value of 0. So, for a function with a single output, the \(EmaX\) algorithm attempts to maximize \(E[y_{\text{min}}^n - Y(x)|Y^n = y^n]\), which is equivalent to minimizing \(E[Y(x)|Y^n = y^n]\). So, for a function with a single output, the \(EmaX\) algorithm myopically searches for the minimum of the predicted response surface \(\hat{y}(x) = E[Y(x)|Y^n = y^n]\), while ignoring the uncertainty of this predicted surface. Essentially, it would lack the balance between exploitation and exploration that makes the expected improvement algorithm introduced in Schonlau (1997) and Jones et al. (1998) so appealing. Unless the
initial design $D_n$ was “large” enough so that $\hat{y}(x)$ is a good approximation to $y(x)$, such an optimization routine could be problematic.

3. **Scale Invariance**: The Pareto improvement function depends on the scaling of the output. Bautista (2009) handled this by scaling each of the outputs so that the largest observed value in each dimension corresponds to 1, and the smallest observed output in each dimension corresponds to 0.

4. **Free of External Parameters** The Pareto improvement function does not depend on any external parameters.

5. **Relationship to a Pareto Set Quality Indicator**: While not explicitly related to a Pareto set approximation quality indicator, the Pareto improvement function is related to the maximin fitness function, which Balling (2003) derived directly from the notion of Pareto dominance. Moreover, if we “truncate” $I_P(\cdot)$ so that it always evaluates to 0 for dominated solutions, we can show an interesting relationship to the binary-ε indicator, a Pareto set approximation quality indicator described in Zitzler et al. (2003). This relationship is discussed earlier in Section 2.3.1.

6. **Exact Computability**: The maximin fitness function, and thus the Pareto improvement function, is relatively quick to compute, and we do not need to resort to any approximations in its computation. Balling (2003) cites its efficiency and simple implementation of the maximin fitness function as its main strengths.
\(I_M(\cdot):\) Maximin Improvement

1. **Monotonicity:** \(I_M(\cdot)\) is monotonic with respect to Pareto dominance. To prove this, let \(y(x)\) and \(y(x')\) be two objective space vectors such that \(y(x) \preceq y(x')\). We have

\[
I_M(y(x)) = -\max_{x_i \in P^n_X} \min_{j=1, \ldots, m} (y_j(x) - y_j(x_i)) 1 \left[ -\max_{x_i \in P^n_X} \min_{j=1, \ldots, m} (y_j(x) - y_j(x_i)) < 0 \right]
\]

\[
= \min_{x_i \in P^n_X} \max_{j=1, \ldots, m} (y_j(x_i) - y_j(x)) 1 \left[ -\max_{x_i \in P^n_X} \min_{j=1, \ldots, m} (y_j(x) - y_j(x_i)) < 0 \right]
\]

\[
\leq \min_{x_i \in P^n_X} \max_{j=1, \ldots, m} (y_j(x_i) - y_j(x')) 1 \left[ -\max_{x_i \in P^n_X} \min_{j=1, \ldots, m} (y_j(x') - y_j(x_i)) < 0 \right]
\]

\[
= -\max_{x_i \in P^n_X} \min_{j=1, \ldots, m} (y_j(x') - y_j(x_i)) 1 \left[ -\max_{x_i \in P^n_X} \min_{j=1, \ldots, m} (y_j(x') - y_j(x_i)) < 0 \right]
\]

\[
= I_M(y(x')).
\]

2. **Generalization of \(I(x)\):** Notice that for \(m = 1\),

\[
I_M(y(x)) = -\max_{x_i \in P^n_X} (y(x) - y(x_i)) 1 \left[ -\max_{x_i \in P^n_X} \min_{j=1, \ldots, m} (y_j(x) - y_j(x_i)) < 0 \right]
\]

\[
= \min_{x_i \in P^n_X} (y(x_i) - y(x)) 1 \left[ \max_{x_i \in P^n_X} \min_{j=1, \ldots, m} (y_j(x_i)) - y_j(x) > 0 \right]
\]

\[
= \left( y^n_{\min} - y(x) \right) 1_{[y^n_{\min} - y(x) < 0]}
\]

\[
= I(x)
\]

3. **Scale Invariance:** The Maximin improvement function, just like the Pareto improvement function, depends on the scaling of the output. Just like Bautista (2009), we handle this in our implementation by scaling each of the outputs so that the largest observed value in each dimension corresponds to 1, and the smallest observed output in each dimension corresponds to 0.
4. **Free of External Parameters**: The maximin improvement function does not depend on external parameters.

5. **Relationship to a Pareto Set Quality Indicator**: Theorem 4 shows that the maximin improvement function is equivalent to the additive binary-$\epsilon$ indicator of $P_n^y$ and the set of nondominated outputs if the original design is augmented by $y(x)$.

6. **Exact Computability**: The maximin fitness function, and thus the Maximin improvement function, is relatively quick to compute, and we need not resort to any approximations in its computation.

$I_{WH}(\cdot)$: **Gaussian Weighted Hypervolume Improvement**

1. **Monotonicity**: Consider the case where $y(x) \succeq y(x')$ and $y(x), y(x') \in \mathcal{R}_n$. Then,

$$\{ v \in \mathbb{R}^m \mid y(x') \succeq v, v \in \mathcal{R}_n \} \subseteq \{ v \mathbb{R}^m \mid y(x) \succeq v, v \in \mathcal{R}_n \} ,$$

so that

$$P(y(x) \succeq V, V \in \mathcal{R}_n) \geq P(y(x') \succeq V, V \in \mathcal{R}_n)$$

for $V \sim N(0_m, \zeta^2 I_m)$. Therefore, $I_{WH}(\cdot)$ is monotonic with respect to Pareto dominance. Monotonicity hold trivially when both $y(x)$ and $y(x')$ are dominated by the current Pareto front and when $y(x) \in \mathcal{R}_n$ but $y(x') \notin \mathcal{R}_n$. 

103
2. **Generalization of** $I(x)$: If $m = 1$, then $V \sim N(0, \zeta^2)$ and

$$I_{WH}(y(x)) = P(y(x) \leq V, V \leq y_{min}^n)$$

$$= \int_{y_{min}^n}^{y(x)} \frac{1}{\zeta} \frac{v}{\zeta} dv \left[ y(x) < y_{min}^n \right]$$

$$= \left[ \Phi \left( \frac{y_{min}^n}{\zeta} \right) - \Phi \left( \frac{y(x)}{\zeta} \right) \right] \left[ y(x) < y_{min}^n \right]$$

So, in this sense, $I_{WH}(\cdot)$ cannot be viewed as a multivariate generalization of $I(x)$.

3. **Scale Invariance**: Our definition of $I_{WH}(\cdot)$ in Section 2.3.2 depended upon an empirical scaling of the outputs because $I_{WH}(\cdot)$ is not scaling invariant.

4. **Free of External Parameters**: While we no longer have to choose a dominated reference point $R$, as when using the standard hypervolume improvement function, we now must somehow specify the parameter $\zeta$ (the standard deviation of $V$). So, $I_{WH}(\cdot)$ depends on an external parameter.

5. **Relationship to a Pareto Set Quality Indicator**: The improvement function $I_{WH}(\cdot)$ is explicitly defined in terms of the weighted hypervolume indicator, which is discussed in Section 1.5.2

6. **Exact Computability**: Exact Computability of the Gaussian weighted hypervolume improvement is not possible, as one is required to compute a multivariate normal probability in a somewhat oddly shaped region. We must rely on a Monte Carlo approximation to compute the desired probability.
$I_C(\cdot)$: Completeness Indicator Improvement

1. **Monotonicity**: Consider the case where $y(x) \succeq y(x')$ and $y(x), y(x') \in \mathcal{R}_n$. Then,

$$\{ u \in \mathcal{X} \mid y(x') \succeq y(u), y(u) \in \mathcal{R}_n \} \subseteq \{ u \in \mathcal{X} \mid y(x) \succeq y(u), y(u) \in \mathcal{R}_n \},$$

so that

$$P(y(x) \succeq y(U), y(U) \in \mathcal{R}_n) \geq P(y(x') \succeq y(U), y(U) \in \mathcal{R}_n)$$

for $U$ uniformly distributed over $\mathcal{X}$. Therefore, $I_C(\cdot)$ is monotonic with respect to Pareto dominance. Monotonicity hold trivially when both $y(x)$ and $y(x')$ are dominated by the current Pareto front and when $y(x) \in \mathcal{R}_n$ but $y(x') \not\in \mathcal{R}_n$.

2. **Generalization of $I(x)$**: If $m = 1$, $U$ is uniformly distributed over $\mathcal{X}$, and

$$|\mathcal{X}| = \int_{\mathcal{X}} du$$

we have

$$I_C(y(x)) = P(y(x) \leq y(U), y(U) \leq y_{min}^n)$$

$$= \int_{\mathcal{X}} 1[y(x) \leq y(u), y(u) \leq y_{min}^n] \frac{1}{|\mathcal{X}|} du$$

$$\neq I(x)$$

So, in this sense, $I_C(\cdot)$ cannot be viewed as a multivariate generalization of $I(x)$.

3. **Scale Invariance**: Since $I_C(\cdot)$ is simply defined as the difference of the completeness indicator functions and the completeness indicator is scaling invariant, it follows that $I_C(\cdot)$ is also scaling invariant.

4. **Free of External Parameters**: $I_C(\cdot)$ does not depend on external parameters.

105
5. **Relationship to a Pareto Set Quality Indicator**: The improvement function \( I_C(\cdot) \) is explicitly defined in terms of the completeness indicator, which is discussed in Section 1.5.2.

6. **Exact Computability**: Exact computability of the completeness indicator improvement only occurs if \( y(\cdot) \) is analytically tractable and the integral

\[
\int_{\mathcal{X}} 1_{[y(\mathbf{x}) \geq y(\mathbf{u}), y(\mathbf{u}) \in \mathbb{R}^n]} \frac{1}{|\mathcal{X}|} d\mathbf{u}
\]

is expressed in closed form. In general, this is not the case, and it certainly is not the case in the typical computer experiments setting. Therefore, we must rely on an approximation to compute \( I_C(\cdot) \).

### 2.4.3 Summary and Discussion of Properties

A summary of the properties of the seven improvement functions discussed in this chapter is stated in Table 2.1. Judging solely by these properties, there are several conclusions one can make regarding the different improvement functions. First, none of the improvement functions presented possesses all of the six properties. So, in some sense, the choice of improvement function will depend upon which properties one deems most important. However, the proposed maximin improvement function is the only one that attains five of the six desirable properties. (The only property is does not attain is scaling invariance.) Second, it would appear the the Gaussian weighted hypervolume improvement is the “worst”, as it attains only two of the six properties. However, merely counting the number of properties attained might be unsatisfactory, as one might judge certain properties to be more important than others. For example, if one deems scale invariance to be more important than any other
Improvement

<table>
<thead>
<tr>
<th>Improvement Function</th>
<th>Monotonicity</th>
<th>Generalization of $I(x)$</th>
<th>Scaling Invariance</th>
<th>Free of External Parameters</th>
<th>Relationship to a Pareto Set Quality Indicator</th>
<th>Exact Computation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_P(x)$</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>$I_K(x)$</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>$I_H(x)$</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>$I_F(x)$</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>$I_M(x)$</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>$I_W(x)$</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>$I_C(x)$</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 2.1: Summary of Desirable Properties for existing and proposed improvement functions

property, then one might consider the 0-1 improvement function $I_P(x)$ and the completeness improvement function $I_C(x)$ to be superior to the maximin improvement, as they possess the most indispensable property. On the same token, one might consider the relationship to a Pareto set quality indicator, and the generalization of $I(x)$ to be rather esoteric properties with little real-world applicability, and completely dismiss these properties when considering which improvement function to use.

While the importance of most of these properties is debatable, there is one property that appears essential. This property is monotonicity with respect to Pareto dominance; an improvement function lacking this property would essentially view certain dominated outputs as being better than nondominated ones. In one dimension, this would be akin to saying that there are some large outputs that offer less improvement than small ones. Certainly, the notion that an output of 5 offers more improvement than an output of 2 is preposterous in a minimization problem! For this reason, we consider the improvement function $I_K(x)$ to be suspect, because it does not even respect the dominance relationship in $\mathbb{R}^m$. 

107
While the preceding analysis of the properties of improvement functions can give us some idea as to which are more attractive than others, the real test of their effectiveness is their ability to approximate the Pareto set and front on an actual multiobjective optimization problems. Therefore, in the next chapter, we will use all seven improvement functions on both real-world applications and test problems from the multiobjective optimization literature to see how well they perform. These results, combined with the comparison of desirable properties, will allow us to make our final conclusions regarding which improvement functions should be used, and which should be avoided.
In Chapter 2, we introduced the expected improvement approach to multiobjective optimization of computer experiments. We overviewed several improvement functions from the literature, introduced new improvement functions, and compared various improvement functions based on the attainability of several desirable properties. Additionally, we also reviewed two possible ways (QIM1 and QIM2) one could use these improvement functions to construct a quality of improvement measure $QI(x)$ (also referred to as the expected improvement). In this chapter, we will attempt to judge the quality of the various combinations of improvement functions, quality improvement measure approaches, and GP modeling assumptions based on their ability to approximate the Pareto front and set in various example problems and applications. Recommendations based on these examples and applications will be presented in Chapter 5.
3.1 Examples

In this section, we will use the various expected improvement approaches described in Chapter 2 to solve several test problems, most of which come from the multiobjective optimization literature. Formally counting all covariance choices (nonseparable LMC vs. independence), improvement functions, and definitions of $QI(x)$ (QIM1 vs. QIM2), there are a total of 28 possible ways one could implement an expected improvement algorithm for a multiobjective optimization problem. However, some combinations are redundant, as the QIM2 implementation of the maximin improvement function $I_M(\cdot)$ (see Section 2.3.1) and the Pareto improvement function $I_P(\cdot)$ (see Section 2.2.4) are identical, and the QIM1 and QIM2 implementations of the 0-1 improvement function $I_{PI}(\cdot)$ (see Section 2.2.1) are also identical. Additionally, the QIM1 implementation of the completeness indicator improvement function $I_C(\cdot)$ (see Section 2.3.3) for a dependence model has proven to be impractical, as it demands a large number of multivariate normal probabilities over oddly shaped regions. Therefore, we will only be examining 23 different combinations in these examples.

Abbreviated notation for each particular approach is listed in Table 3.1. In all examples, we will look at two different initial designs. First, we will start with a maximin-LHD with 5 inputs per input dimension (i.e., an $n = 10$ point design if $d = 2$), which is fewer than the recommended “10-per-dimension” rule advocated in Loepky et al. (2009). Second, we will start with a maximin-LHD with 10 inputs per input dimension (i.e., an $n = 20$ point design if $d = 2$), which follows the recommended “10-per-dimension” rule-of-thumb. In both cases, we will sequentially augment the design with ten additional function evaluations.
In the examples below, the initial space-filling maximin LHD constructed using the MATLAB function `bestlh`, available in on-line supplementary material for Forrester et al. (2008). In the case of independent outputs, REML estimates of the covariance parameters (and then \( \hat{y}(x) \), and \( S(x) \)) were obtained using MPErK software (see Han et al. (2011)). In the nonseparable dependence model, the MATLAB function `ga`, also available as a component of the on-line supplementary material to Forrester et al. (2008), was used to obtain the initial estimate of \( \theta \) and \( A \); these values were taken to be the initial point in an application of the MATLAB `fmincon` function to produce the final estimates of \( \theta \) and \( A \). These estimates are then used to calculate \( \hat{y}(x) \), and \( S(x) \). The MATLAB function `paretoset` (see Cao (2007)) was used to calculate \( P_n^{\mathcal{X}} \) and \( P_n^{\mathcal{Y}} \).

NOMADm (see Abramson (2010)), a MATLAB implementation of a mesh adaptive direct search (MADS) algorithm (see Audet and Dennis (2006)), was used to optimize \( QI(x) \). When \( m = 2 \) and the expected maximin improvement function is defined by QIM1, \( QI(x) \) can be directly calculated and optimized in a straightforward manner. When \( m \geq 3 \) or an improvement function other than \( I_{M}(\cdot) \) is used, \( QI(x) \) is optimized via sample average approximation (SAA, described in Shapiro (2003) and Section 1.6). See Section 2.2 and Section 2.3 for more implementation details for the specific approaches.

In all examples, we will use two quality indicators. To compare the estimated Pareto fronts based on final designs of size \( n^* \) (initial plus sequentially added points), we will calculate \( I_H(P^{n*}_Y) \) using the \( m \) marginal maxima as a reference point. To ease comparisons, we will scale all outputs and reference points so that they are inside the
Table 3.1: List of the expected improvement algorithms used to approximate the Pareto front. Notice that “Method” (QIM1 or QIM2) is not specified for the probability of improvement, as QIM1 and QIM2 are equivalent. Also, “Dependent” refers to a nonseparable LMC model (see Section 1.3.2).

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>$I^*(\cdot)$ (Section)</th>
<th>$QI(\cdot)$</th>
<th>GP Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMMI-Ind</td>
<td>$I_M(\cdot)$</td>
<td>QIM1</td>
<td>Independent</td>
</tr>
<tr>
<td>KEI-Ind</td>
<td>$I_K(\cdot)$</td>
<td>QIM2</td>
<td>Independent</td>
</tr>
<tr>
<td>PI-Ind</td>
<td>$I_T(\cdot)$</td>
<td>-</td>
<td>Independent</td>
</tr>
<tr>
<td>EMAX-Ind</td>
<td>$I_F(\cdot)$</td>
<td>QIM1</td>
<td>Independent</td>
</tr>
<tr>
<td>ECI1-Ind</td>
<td>$I_C(\cdot)$</td>
<td>QIM1 (Averaged)</td>
<td>Independent</td>
</tr>
<tr>
<td>ECI2-Ind</td>
<td>$I_C(\cdot)$</td>
<td>QIM2 (Averaged)</td>
<td>Independent</td>
</tr>
<tr>
<td>EHI1-Ind</td>
<td>$I_H(\cdot)$</td>
<td>QIM1</td>
<td>Independent</td>
</tr>
<tr>
<td>EHI2-Ind</td>
<td>$I_H(\cdot)$</td>
<td>QIM2</td>
<td>Independent</td>
</tr>
<tr>
<td>EWH1-Ind</td>
<td>$I_W(\cdot)$</td>
<td>QIM1</td>
<td>Independent</td>
</tr>
<tr>
<td>EWH2-Ind</td>
<td>$I_W(\cdot)$</td>
<td>QIM2</td>
<td>Independent</td>
</tr>
<tr>
<td>EMMI2-Ind</td>
<td>$I_M(\cdot)$</td>
<td>QIM2</td>
<td>Independent</td>
</tr>
<tr>
<td>KEI1-Ind</td>
<td>$I_K(\cdot)$</td>
<td>QIM1</td>
<td>Independent</td>
</tr>
<tr>
<td>EECEI1-Ind</td>
<td>$I_C(\cdot)$</td>
<td>QIM1 (Estimated)</td>
<td>Independent</td>
</tr>
<tr>
<td>EECEI2-Ind</td>
<td>$I_C(\cdot)$</td>
<td>QIM2 (Estimated)</td>
<td>Independent</td>
</tr>
<tr>
<td>EMMI-Dep</td>
<td>$I_M(\cdot)$</td>
<td>QIM1</td>
<td>Dependent</td>
</tr>
<tr>
<td>KEI-Dep</td>
<td>$I_K(\cdot)$</td>
<td>QIM2</td>
<td>Dependent</td>
</tr>
<tr>
<td>PI-Dep</td>
<td>$I_T(\cdot)$</td>
<td>-</td>
<td>Dependent</td>
</tr>
<tr>
<td>EMAX-Dep</td>
<td>$I_F(\cdot)$</td>
<td>QIM1</td>
<td>Dependent</td>
</tr>
<tr>
<td>ECI2-Dep</td>
<td>$I_C(\cdot)$</td>
<td>QIM2 (Averaged)</td>
<td>Dependent</td>
</tr>
<tr>
<td>EHI1-Dep</td>
<td>$I_H(\cdot)$</td>
<td>QIM1</td>
<td>Dependent</td>
</tr>
<tr>
<td>EHI2-Dep</td>
<td>$I_H(\cdot)$</td>
<td>QIM2</td>
<td>Dependent</td>
</tr>
<tr>
<td>EWH1-Dep</td>
<td>$I_W(\cdot)$</td>
<td>QIM1</td>
<td>Dependent</td>
</tr>
<tr>
<td>EWH2-Dep</td>
<td>$I_W(\cdot)$</td>
<td>QIM2</td>
<td>Dependent</td>
</tr>
<tr>
<td>EMMI2-Dep</td>
<td>$I_M(\cdot)$</td>
<td>QIM2</td>
<td>Dependent</td>
</tr>
<tr>
<td>KEI1-Dep</td>
<td>$I_K(\cdot)$</td>
<td>QIM1</td>
<td>Dependent</td>
</tr>
<tr>
<td>EECEI1-Dep</td>
<td>$I_C(\cdot)$</td>
<td>QIM1 (Estimated)</td>
<td>Dependent</td>
</tr>
<tr>
<td>EECEI2-Dep</td>
<td>$I_C(\cdot)$</td>
<td>QIM2 (Estimated)</td>
<td>Dependent</td>
</tr>
</tbody>
</table>
The hypercube $[0,1]^m$, with 0 corresponding to the marginal minima and 1 corresponding to the marginal maxima. (Larger values of the hypervolume indicator represent higher quality Pareto fronts.) Additionally, after the aforementioned scaling, we will also calculate $I_{e^+}(\mathcal{P}_y, \mathcal{P}_y^*)$. (Recall, smaller values of the binary-$\epsilon$ indicator represent higher quality Pareto fronts.)

In the cases where there is two dimensional input and/or two dimensional output, we will also use some graphical methods to assess the the quality of the sequential design created via the various expected improvement algorithms. Specifically, in cases where we have created five sequential designs for each expected improvement algorithm, we will plot the sequentially added inputs and the original design of the final run of the expected improvement algorithm. Also, we will plot the outputs corresponding to these sequentially added inputs and the outputs at the original design of the final run of the expected improvement algorithm. In cases where there is only one expected improvement algorithm is run, we will simply plot the results of the single run.

### 3.1.1 MOP2

The first example considered is commonly referred to as the MOP2 problem; this test problem was first described in Fonseca and Fleming (1995). MOP2 has a $d = 2$-dimensional input space $\mathcal{X} = [-2,2]^2$, and $m = 2$ objective functions which are

\[
y_1(x) = 1 - \exp \left\{ -\sum_{i=1}^{2} \left( x_i - \frac{1}{\sqrt{2}} \right)^2 \right\}, \quad \text{and} \quad (3.1.1)
\]

\[
y_2(x) = 1 - \exp \left\{ -\sum_{i=1}^{2} \left( x_i + \frac{1}{\sqrt{2}} \right)^2 \right\}. \quad (3.1.2)
\]
The Pareto set is the line segment

\[ \mathcal{P}_X = \left\{ \mathbf{x} : x_1 = x_2 \text{ and } -\frac{1}{\sqrt{2}} \leq x_1 \leq \frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}} \leq x_2 \leq \frac{1}{\sqrt{2}} \right\}. \]

A discrete approximation to \( \mathcal{P}_Y \) was determined by evaluating \((y_1(x), y_2(x))\) at 201 \( \mathbf{x} \) points uniformly spread in \( \mathcal{P}_X \). This close approximation to \( \mathcal{P}_Y \) served as the basis for comparing the various Pareto front approximations constructed for this example.

From the results in Table 3.2, where an initial 10 point design is proceeded by 10 sequentially added inputs, in terms of the hypervolume indicator, it appears that most methods based on the modified maximin fitness function (EMMI-Ind, EMMI2-Ind, EMAX-Ind, EMMI-Dep and EMAX-Dep), hypervolume based methods (EHI1-Ind, EHI2-Ind, EHI1-Dep, and EHI2-Dep), and ECI2-Ind performed substantially better than the other implementations using either the dependent or independence process model for \((Y_1(x), Y_2(x))\). The ten aforementioned approaches all have have average hypervolume measures greater than 0.286. All scaling invariant methods other than ECI2-Ind, ECI2-Dep, EECI1-Ind, and EECI2-Dep (PI-Ind, PI-Dep, EECI2-Ind,
EECI2-Dep and ECI1-Ind) are performing the worst in terms of the hypervolume, with ECI1-Ind and both EECI2 implementations performing far worse than any other method.

In terms of the binary-\(\epsilon\) indicator results in Table 3.2, the methods based on the modified maximin fitness function appear to perform best, but the average binary-\(\epsilon\) indicator value for ECI2-Ind is only slightly larger, on average, than the maximin fitness based approaches. The six maximin fitness based methods plus ECI2 are the only approaches with binary-\(\epsilon\) indicators less than 0.08 on average. Again, with the exception of both ECI2 and both EECI1 implementation, the scaling invariant methods are greatly outperformed, with ECI1 and both EECI2 implementation being the poorest of all methods.

In this \(m = 2\) example, one can also visually examine the estimated Pareto front and set. For the most part, Figures 3.5 - 3.14 support the superiority of ECI2,
Table 3.2: Summary of quality indicators for five runs of each algorithm for the *MOP2* problem. The initial design is a 10-point maximin LHD, and 10 points are added via the expected improvement algorithm.

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Min</th>
<th>Max</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMMI-Ind</td>
<td>0.2887</td>
<td>0.0008</td>
<td>0.2876</td>
<td>0.2897</td>
<td>0.0717</td>
<td>0.0038</td>
<td>0.0677</td>
<td>0.0758</td>
</tr>
<tr>
<td>KEI-Ind</td>
<td>0.2694</td>
<td>0.0103</td>
<td>0.2529</td>
<td>0.2800</td>
<td>0.0957</td>
<td>0.0102</td>
<td>0.0841</td>
<td>0.1117</td>
</tr>
<tr>
<td>PI-Ind</td>
<td>0.2520</td>
<td>0.0144</td>
<td>0.2342</td>
<td>0.2688</td>
<td>0.1190</td>
<td>0.0169</td>
<td>0.1043</td>
<td>0.1461</td>
</tr>
<tr>
<td>EMAX-Ind</td>
<td>0.2884</td>
<td>0.0013</td>
<td>0.2862</td>
<td>0.2895</td>
<td>0.0721</td>
<td>0.0043</td>
<td>0.0680</td>
<td>0.0788</td>
</tr>
<tr>
<td>ECI1-Ind</td>
<td>0.0863</td>
<td>0.0132</td>
<td>0.0734</td>
<td>0.1068</td>
<td>0.2877</td>
<td>0.0162</td>
<td>0.2588</td>
<td>0.2950</td>
</tr>
<tr>
<td>ECI2-Ind</td>
<td>0.2887</td>
<td>0.0014</td>
<td>0.2874</td>
<td>0.2909</td>
<td>0.0765</td>
<td>0.0095</td>
<td>0.0666</td>
<td>0.0879</td>
</tr>
<tr>
<td>EHI1-Ind</td>
<td>0.2908</td>
<td>0.0062</td>
<td>0.2802</td>
<td>0.2961</td>
<td>0.0893</td>
<td>0.0054</td>
<td>0.0803</td>
<td>0.0927</td>
</tr>
<tr>
<td>EHI2-Ind</td>
<td>0.2878</td>
<td>0.0033</td>
<td>0.2824</td>
<td>0.2905</td>
<td>0.1089</td>
<td>0.0329</td>
<td>0.0785</td>
<td>0.1615</td>
</tr>
<tr>
<td>EWHI1-Ind</td>
<td>0.2812</td>
<td>0.0018</td>
<td>0.2794</td>
<td>0.2839</td>
<td>0.0849</td>
<td>0.0047</td>
<td>0.0780</td>
<td>0.0896</td>
</tr>
<tr>
<td>EWHI2-Ind</td>
<td>0.2556</td>
<td>0.0099</td>
<td>0.2457</td>
<td>0.2665</td>
<td>0.1194</td>
<td>0.0224</td>
<td>0.0927</td>
<td>0.1444</td>
</tr>
<tr>
<td>EMMI2-Ind</td>
<td>0.2863</td>
<td>0.0010</td>
<td>0.2848</td>
<td>0.2874</td>
<td>0.0709</td>
<td>0.0015</td>
<td>0.0690</td>
<td>0.0729</td>
</tr>
<tr>
<td>KEI1-Ind</td>
<td>0.2731</td>
<td>0.0014</td>
<td>0.2712</td>
<td>0.2746</td>
<td>0.0938</td>
<td>0.0016</td>
<td>0.0927</td>
<td>0.0958</td>
</tr>
<tr>
<td>EECI1-Ind</td>
<td>0.2820</td>
<td>0.0021</td>
<td>0.2796</td>
<td>0.2853</td>
<td>0.0945</td>
<td>0.0162</td>
<td>0.0675</td>
<td>0.1077</td>
</tr>
<tr>
<td>EECI2-Ind</td>
<td>0.0697</td>
<td>0.0022</td>
<td>0.0684</td>
<td>0.0734</td>
<td>0.3773</td>
<td>0.0751</td>
<td>0.2950</td>
<td>0.4321</td>
</tr>
<tr>
<td>EMMI-Dep</td>
<td>0.2880</td>
<td>0.0011</td>
<td>0.2860</td>
<td>0.2888</td>
<td>0.0742</td>
<td>0.0016</td>
<td>0.0717</td>
<td>0.0758</td>
</tr>
<tr>
<td>KEI-Dep</td>
<td>0.2681</td>
<td>0.0051</td>
<td>0.2621</td>
<td>0.2744</td>
<td>0.0905</td>
<td>0.0056</td>
<td>0.0830</td>
<td>0.0967</td>
</tr>
<tr>
<td>PI-Dep</td>
<td>0.2370</td>
<td>0.0167</td>
<td>0.2196</td>
<td>0.2582</td>
<td>0.1323</td>
<td>0.0290</td>
<td>0.1085</td>
<td>0.1824</td>
</tr>
<tr>
<td>EMAX-Dep</td>
<td>0.2860</td>
<td>0.0041</td>
<td>0.2792</td>
<td>0.2894</td>
<td>0.0740</td>
<td>0.0041</td>
<td>0.0689</td>
<td>0.0797</td>
</tr>
<tr>
<td>ECI2-Dep</td>
<td>0.2824</td>
<td>0.0063</td>
<td>0.2732</td>
<td>0.2889</td>
<td>0.0886</td>
<td>0.0076</td>
<td>0.0797</td>
<td>0.0982</td>
</tr>
<tr>
<td>EHI1-Dep</td>
<td>0.2901</td>
<td>0.0038</td>
<td>0.2858</td>
<td>0.2955</td>
<td>0.0890</td>
<td>0.0133</td>
<td>0.0662</td>
<td>0.1009</td>
</tr>
<tr>
<td>EHI2-Dep</td>
<td>0.2869</td>
<td>0.0034</td>
<td>0.2828</td>
<td>0.2913</td>
<td>0.0875</td>
<td>0.0219</td>
<td>0.0660</td>
<td>0.1184</td>
</tr>
<tr>
<td>EWHI1-Dep</td>
<td>0.2788</td>
<td>0.0057</td>
<td>0.2709</td>
<td>0.2862</td>
<td>0.0909</td>
<td>0.0094</td>
<td>0.0792</td>
<td>0.1044</td>
</tr>
<tr>
<td>EWHI2-Dep</td>
<td>0.2581</td>
<td>0.0175</td>
<td>0.2297</td>
<td>0.2758</td>
<td>0.1063</td>
<td>0.0222</td>
<td>0.0805</td>
<td>0.1387</td>
</tr>
<tr>
<td>EMMI2-Dep</td>
<td>0.2823</td>
<td>0.0041</td>
<td>0.2777</td>
<td>0.2866</td>
<td>0.0780</td>
<td>0.0083</td>
<td>0.0721</td>
<td>0.0927</td>
</tr>
<tr>
<td>KEI1-Dep</td>
<td>0.2686</td>
<td>0.0047</td>
<td>0.2629</td>
<td>0.2748</td>
<td>0.0889</td>
<td>0.0061</td>
<td>0.0790</td>
<td>0.0930</td>
</tr>
<tr>
<td>EECI1-Dep</td>
<td>0.2806</td>
<td>0.0059</td>
<td>0.2724</td>
<td>0.2886</td>
<td>0.0953</td>
<td>0.0129</td>
<td>0.0758</td>
<td>0.1094</td>
</tr>
<tr>
<td>EECI2-Dep</td>
<td>0.0685</td>
<td>0.0000</td>
<td>0.0684</td>
<td>0.0685</td>
<td>0.4321</td>
<td>0.0000</td>
<td>0.4321</td>
<td>0.4321</td>
</tr>
</tbody>
</table>
Table 3.3: Summary of quality indicators for five runs of each algorithm for the MOP2 problem. The initial design is a 20-point maximin LHD, and 10 points are added via the expected improvement algorithm.

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Min</th>
<th>Max</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMMI-Ind</td>
<td>0.2906</td>
<td>0.0004</td>
<td>0.2899</td>
<td>0.2911</td>
<td>0.0641</td>
<td>0.0025</td>
<td>0.0619</td>
<td>0.0674</td>
</tr>
<tr>
<td>KEI-Ind</td>
<td>0.2711</td>
<td>0.0041</td>
<td>0.2650</td>
<td>0.2760</td>
<td>0.0837</td>
<td>0.0070</td>
<td>0.0754</td>
<td>0.0902</td>
</tr>
<tr>
<td>PI-Ind</td>
<td>0.2519</td>
<td>0.0055</td>
<td>0.2456</td>
<td>0.2585</td>
<td>0.1136</td>
<td>0.0127</td>
<td>0.0978</td>
<td>0.1301</td>
</tr>
<tr>
<td>EMAX1-Ind</td>
<td>0.2898</td>
<td>0.0011</td>
<td>0.2883</td>
<td>0.2909</td>
<td>0.0690</td>
<td>0.0037</td>
<td>0.0671</td>
<td>0.0757</td>
</tr>
<tr>
<td>ECI1-Ind</td>
<td>0.1790</td>
<td>0.0035</td>
<td>0.1772</td>
<td>0.1852</td>
<td>0.1812</td>
<td>0.0000</td>
<td>0.1812</td>
<td>0.1812</td>
</tr>
<tr>
<td>ECI2-Ind</td>
<td>0.2864</td>
<td>0.0021</td>
<td>0.2839</td>
<td>0.2895</td>
<td>0.0882</td>
<td>0.0106</td>
<td>0.0739</td>
<td>0.0964</td>
</tr>
<tr>
<td>EHI1-Ind</td>
<td>0.2941</td>
<td>0.0022</td>
<td>0.2905</td>
<td>0.2962</td>
<td>0.1189</td>
<td>0.0328</td>
<td>0.0821</td>
<td>0.1528</td>
</tr>
<tr>
<td>EHI2-Ind</td>
<td>0.2938</td>
<td>0.0010</td>
<td>0.2926</td>
<td>0.2951</td>
<td>0.1037</td>
<td>0.0423</td>
<td>0.0583</td>
<td>0.1528</td>
</tr>
<tr>
<td>EWHI1-Ind</td>
<td>0.2830</td>
<td>0.0050</td>
<td>0.2775</td>
<td>0.2884</td>
<td>0.0843</td>
<td>0.0111</td>
<td>0.0719</td>
<td>0.0987</td>
</tr>
<tr>
<td>EWHI2-Ind</td>
<td>0.2782</td>
<td>0.0042</td>
<td>0.2712</td>
<td>0.2821</td>
<td>0.0856</td>
<td>0.0149</td>
<td>0.0708</td>
<td>0.1084</td>
</tr>
<tr>
<td>EMMI2-Ind</td>
<td>0.2891</td>
<td>0.0020</td>
<td>0.2874</td>
<td>0.2917</td>
<td>0.0641</td>
<td>0.0021</td>
<td>0.0618</td>
<td>0.0669</td>
</tr>
<tr>
<td>KEI1-Ind</td>
<td>0.2687</td>
<td>0.0047</td>
<td>0.2640</td>
<td>0.2748</td>
<td>0.0871</td>
<td>0.0049</td>
<td>0.0792</td>
<td>0.0920</td>
</tr>
<tr>
<td>EECP1-Ind</td>
<td>0.2858</td>
<td>0.0030</td>
<td>0.2834</td>
<td>0.2905</td>
<td>0.0771</td>
<td>0.0017</td>
<td>0.0742</td>
<td>0.0788</td>
</tr>
<tr>
<td>EECP2-Ind</td>
<td>0.1738</td>
<td>0.0000</td>
<td>0.1738</td>
<td>0.1738</td>
<td>0.1812</td>
<td>0.0000</td>
<td>0.1812</td>
<td>0.1812</td>
</tr>
<tr>
<td>EMMI-Dep</td>
<td>0.2906</td>
<td>0.0006</td>
<td>0.2899</td>
<td>0.2913</td>
<td>0.0649</td>
<td>0.0028</td>
<td>0.0619</td>
<td>0.0674</td>
</tr>
<tr>
<td>KEI-Dep</td>
<td>0.2605</td>
<td>0.0123</td>
<td>0.2479</td>
<td>0.2766</td>
<td>0.1003</td>
<td>0.0156</td>
<td>0.0763</td>
<td>0.1130</td>
</tr>
<tr>
<td>PI-Dep</td>
<td>0.2559</td>
<td>0.0116</td>
<td>0.2375</td>
<td>0.2646</td>
<td>0.1172</td>
<td>0.0315</td>
<td>0.0860</td>
<td>0.1574</td>
</tr>
<tr>
<td>EMAX1-Dep</td>
<td>0.2894</td>
<td>0.0007</td>
<td>0.2888</td>
<td>0.2902</td>
<td>0.0690</td>
<td>0.0042</td>
<td>0.0668</td>
<td>0.0765</td>
</tr>
<tr>
<td>ECI2-Dep</td>
<td>0.2872</td>
<td>0.0028</td>
<td>0.2832</td>
<td>0.2899</td>
<td>0.0887</td>
<td>0.0111</td>
<td>0.0731</td>
<td>0.0986</td>
</tr>
<tr>
<td>EHI1-Dep</td>
<td>0.2925</td>
<td>0.0028</td>
<td>0.2898</td>
<td>0.2967</td>
<td>0.1243</td>
<td>0.0399</td>
<td>0.0696</td>
<td>0.1528</td>
</tr>
<tr>
<td>EHI2-Dep</td>
<td>0.2890</td>
<td>0.0022</td>
<td>0.2852</td>
<td>0.2910</td>
<td>0.1230</td>
<td>0.0246</td>
<td>0.0956</td>
<td>0.1528</td>
</tr>
<tr>
<td>EWHI1-Dep</td>
<td>0.2852</td>
<td>0.0035</td>
<td>0.2808</td>
<td>0.2888</td>
<td>0.0780</td>
<td>0.0056</td>
<td>0.0713</td>
<td>0.0843</td>
</tr>
<tr>
<td>EWHI2-Dep</td>
<td>0.2761</td>
<td>0.0055</td>
<td>0.2687</td>
<td>0.2828</td>
<td>0.0805</td>
<td>0.0167</td>
<td>0.0662</td>
<td>0.1089</td>
</tr>
<tr>
<td>EMMI2-Dep</td>
<td>0.2880</td>
<td>0.0017</td>
<td>0.2861</td>
<td>0.2894</td>
<td>0.0674</td>
<td>0.0034</td>
<td>0.0641</td>
<td>0.0727</td>
</tr>
<tr>
<td>KEI1-Dep</td>
<td>0.2717</td>
<td>0.0095</td>
<td>0.2609</td>
<td>0.2868</td>
<td>0.0908</td>
<td>0.0186</td>
<td>0.0648</td>
<td>0.1139</td>
</tr>
<tr>
<td>EECP1-Ind</td>
<td>0.2875</td>
<td>0.0013</td>
<td>0.2859</td>
<td>0.2892</td>
<td>0.0777</td>
<td>0.0016</td>
<td>0.0742</td>
<td>0.0781</td>
</tr>
<tr>
<td>EECP2-Ind</td>
<td>0.1738</td>
<td>0.0000</td>
<td>0.1738</td>
<td>0.1738</td>
<td>0.1812</td>
<td>0.0000</td>
<td>0.1812</td>
<td>0.1812</td>
</tr>
</tbody>
</table>
the maximin fitness based methods, and the hypervolume based methods. Among these methods, EMMI2 and EHI2 (with either an independence or dependence model) show more spread in the input space, and are more widely distributed around the true Pareto set. Additionally, these plots show why ECI1-Ind, EECI2-Ind, and EECI2-Dep perform so poorly; it appears that these particular criteria favor regions in the input space where there is a large amount of predictive uncertainty. In particular, ECI1-Ind seems to greatly favor areas outside of the convex hull of the original design, near the boundaries of the input space. Thus, ECI1-Ind seems to overvalue exploration of the input space.

We should note that EECI1-Dep, EECI1-Dep, and the other scaling dependent methods (KEI, KEI1, EWHI1, and EWHI2) produce reasonable results. However, these results are inferior to both ECI2 implementations, the methods based on the maximin improvement, and those based on hypervolume improvement. Using Figures 3.5 - 3.14, it seems that these methods, with the exception of possibly EWHI1-Ind, EWHI2-Ind and EECI1-Ind, all suffer from “over-exploring” the input space and output space, as opposed to effectively zeroing in on the Pareto set and front.

A somewhat surprising result is that the dependence GP model, while not necessarily producing bad results, appears to offer little advantage over the less computationally demanding independence model. On average, it seems to perform slightly worse for almost all improvement criteria in terms of the hypervolume indicator. EWHI2 is the only criterion where we see greater average hypervolume values for the dependence model. There are a few more instances where the binary—$\epsilon$ indicator is smaller for the dependence model, but the differences are small. Visually, judging
by Figures 3.5 - 3.13, it does not appear that any of the dependence models produce better Pareto fronts and sets.

One possible explanation is that the nonseparable dependence GP model does not model this particular function well, and a different dependence model for $Y(x)$ is more appropriate. Another possible explanation is that the particular dependence model is appropriate, but our estimated covariance parameters, at some stages of the sequential design algorithm, are not globally optimal. Recall that obtaining estimates of the covariance parameters for the dependence model requires maximization of a restricted likelihood function of seven parameters. This is computationally more difficult than the parameter estimation in the independence model, which requires us to maximize two separate restricted likelihood functions, each of which depends on only two parameters.

Aside from larger hypervolumes and lower binary$-\epsilon$ indicators (which are to be expected with a larger initial design), the results in Table 3.3, where an initial 20 point design is followed by 10 sequentially added inputs, lead to similar conclusions as Table 3.2. Again, the hypervolume based methods, maximin fitness based methods, and ECI2 seem to be performing the best. Moreover, EECI1-Ind and EECI1-Dep now appear to be even more competitive, and possibly better, than ECI2-Ind and ECI2-Dep. Other scaling invariant methods perform poorly, with ECI1 and EECI2 performing the worst. Figures 3.15 - 3.23 also seem to support the superiority of the hypervolume based methods, maximin fitness based methods, EECI1 and ECI2, while also displaying the inferiority of the other scaling invariant methods and the mediocrity of KEI, KEI1, EWHI1, and EWHI2. The extremely poor performance of ECI1 and EECI2 can, again, be attributed to over-exploration of the input space.
Figure 3.5: The final run of the five sequential designs for the MOP2 problem with an initial design of size 10 and 10 points added sequentially via EMMI-Ind (top), EMAX1-Ind (middle) and EMMI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.6: The final run of the five sequential designs for the MOP2 problem with an initial design of size 10 and 10 points added sequentially via EHI1-Ind (top), EWHI1-Ind (middle) and KEI-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.7: The final run of the five sequential designs for the MOP2 problem with an initial design of size 10 and 10 points added sequentially via EHI2-Ind (top), EWHI2-Ind (middle) and KEI1-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.8: The final run of the five sequential designs for the MOP2 problem with an initial design of size 10 and 10 points added sequentially via PI-Ind (top), ECI1-Ind (middle) and ECI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.9: The final run of the five sequential designs for the MOP2 problem with an initial design of size 10 and 10 points added sequentially via EECI1-Ind (top) and EECI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.10: The final run of the five sequential designs for the MOP2 problem with an initial design of size 10 and 10 points added sequentially via EMMI-Dep (top), EMAX1-Dep (middle) and EMMI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.11: The final run of the five sequential designs for the MOP2 problem with an initial design of size 10 and 10 points added sequentially via EHI1-Dep (top), EWHI1-Dep (middle) and KEI-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.12: The final run of the five sequential designs for the MOP2 problem with an initial design of size 10 and 10 points added sequentially via EHI2-Dep (top), EWHI2-Dep (middle) and KEI1-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.13: The final run of the five sequential designs for the MOP2 problem with an initial design of size 10 and 10 points added sequentially via PI-Dep (top) and ECI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.14: The final run of the five sequential designs for the MOP2 problem with an initial design of size 10 and 10 points added sequentially via EECI1-Dep (top) and EECI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.15: The final run of the five sequential designs for the MOP2 problem with an initial design of size 20 and 10 points added sequentially via EMMI-Ind (top), EMAX1-Ind (middle) and EMMI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.16: The final run of the five sequential designs for the MOP2 problem with an initial design of size 20 and 10 points added sequentially via EHI-Ind (top), EWHI-Ind (middle) and KEI-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.17: The final run of the five sequential designs for the MOP2 problem with an initial design of size 20 and 10 points added sequentially via EHI2-Ind (top), EWHI2-Ind (middle) and KEI1-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.18: The final run of the five sequential designs for the MOP2 problem with an initial design of size 20 and 10 points added sequentially via PI-Ind (top), ECI1-Ind (middle) and ECI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.19: The final run of the five sequential designs for the MOP2 problem with an initial design of size 20 and 10 points added sequentially via EECI1-Ind (top) and EECI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.20: The final run of the five sequential designs for the MOP2 problem with an initial design of size 20 and 10 points added sequentially via EMMI-Dep (top), EMAX1-Dep (middle) and EMMI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.21: The final run of the five sequential designs for the MOP2 problem with an initial design of size 20 and 10 points added sequentially via EHI1-Dep (top), EWHI1-Dep (middle) and KEI-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.22: The final run of the five sequential designs for the MOP2 problem with an initial design of size 20 and 10 points added sequentially via EHI2-Dep (top), EWHI2-Dep (middle) and KEI1-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.23: The final run of the five sequential designs for the MOP2 problem with an initial design of size 20 and 10 points added sequentially via PI-Dep (top) and ECI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.24: The final run of the five sequential designs for the MOP2 problem with an initial design of size 20 and 10 points added sequentially via EECI1-Dep (top) and EECI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
3.1.2 WSNL

The WSNL function was introduced in Williams et al. (2010) as a single-objective optimization problem with one constraint function, two control variables and four environmental variables. Here, we modify this function, as in Bautista (2009), so that it has $d = 2$ inputs and $m = 2$ output functions. The outputs for the WSNL function are

$$y_1(x) = \eta_1(x) \times \frac{1}{3^7} \sum_{i=1}^{3} \sum_{j=1}^{3} \sum_{k=1}^{3} \sum_{l=1}^{3} o_1(a_i, b_j, c_k, d_l)$$

$$y_2(x) = \eta_2(x) \times \frac{1}{3^7} \sum_{i=1}^{3} \sum_{j=1}^{3} \sum_{k=1}^{3} \sum_{l=1}^{3} o_2(a_i, b_j, c_k, d_l),$$

where the components of each function are defined as

$$\eta_1(x) = \left[ x_2 - \frac{5.1x_1^2}{4\pi^2} \frac{5x_1^2}{\pi} - 6 \right] + 10 \left( 1 - \frac{1}{8\pi} \right) \cos(x_1) + 10$$

$$\eta_2(x) = -\sqrt{(10.5 - x_1)(x_1 + 5.5)(x_2 + 0.5) - \left( x_2 - \frac{5.1x_1^2}{4\pi^2} - 6 \right)^2} - \cos(x_1) \left( 1 - \frac{1}{8\pi} \right) - \frac{1}{3}$$
\[ a_1 = b_1 = c_1 = d_1 = 0.25 \]
\[ a_2 = b_2 = c_2 = d_2 = 0.50 \]
\[ a_3 = b_3 = c_3 = d_3 = 0.75 \]

\[ o_1(a_i, b_j, c_k, d_l) = 2(2a_i)^2 + 4.5(2b_j)^{1.5} + 2b_j + 14a_i c_k + 2\sqrt{b_jd_l} \]
\[ o_2(a_i, b_j, c_k, d_l) = 1.2(2b_j)(2a_i)^{1.3} + 4.5(2b_j)^3 + 2(2b_j)^{0.6} + 3.5(4a_i c_k)^{1.7} + (4b_jd_l)^{0.7}. \]

![Figure 3.27](image1.png)  
Figure 3.27: The dark black region is the approximated true Pareto front for the WSNL function

![Figure 3.28](image2.png)  
Figure 3.28: The dark black region is the approximated true Pareto set for the WSNL function

Since there is no analytic form of the Pareto set or front in this particular problem, a discrete approximation to \( \mathcal{P}_x \) and \( \mathcal{P}_y \) was formed by evaluating \( y(x) \) over a 300 × 300 grid in the input space. These approximations are plotted in Figure 3.27 and Figure 3.28, and will be used as the basis of comparison for the various methods for approximating the Pareto front.

141
Table 3.4: Summary of quality indicators for five runs of each algorithm for the WSNL problem. The initial design is a 20-point maximin LHD, and 10 points are added via the expected improvement algorithm.

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Min</th>
<th>Max</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMMI-Ind</td>
<td>0.8863</td>
<td>0.0017</td>
<td>0.8852</td>
<td>0.8892</td>
<td>0.0591</td>
<td>0.0075</td>
<td>0.0461</td>
<td>0.0645</td>
</tr>
<tr>
<td>KEI-Ind</td>
<td>0.8727</td>
<td>0.0077</td>
<td>0.8597</td>
<td>0.8783</td>
<td>0.0811</td>
<td>0.0086</td>
<td>0.0720</td>
<td>0.0939</td>
</tr>
<tr>
<td>PI-Ind</td>
<td>0.8527</td>
<td>0.0109</td>
<td>0.8393</td>
<td>0.8661</td>
<td>0.0776</td>
<td>0.0206</td>
<td>0.0534</td>
<td>0.1054</td>
</tr>
<tr>
<td>EMAX1-Ind</td>
<td>0.8909</td>
<td>0.0020</td>
<td>0.8892</td>
<td>0.8939</td>
<td>0.0452</td>
<td>0.0044</td>
<td>0.0377</td>
<td>0.0492</td>
</tr>
<tr>
<td>ECI1-Ind</td>
<td>0.8632</td>
<td>0.0064</td>
<td>0.8545</td>
<td>0.8717</td>
<td>0.0661</td>
<td>0.0177</td>
<td>0.0520</td>
<td>0.0939</td>
</tr>
<tr>
<td>ECI2-Ind</td>
<td>0.8718</td>
<td>0.0024</td>
<td>0.8693</td>
<td>0.8751</td>
<td>0.0527</td>
<td>0.0066</td>
<td>0.0455</td>
<td>0.0592</td>
</tr>
<tr>
<td>EHI1-Ind</td>
<td>0.8816</td>
<td>0.0052</td>
<td>0.8750</td>
<td>0.8855</td>
<td>0.0491</td>
<td>0.0066</td>
<td>0.0427</td>
<td>0.0596</td>
</tr>
<tr>
<td>EHI2-Ind</td>
<td>0.8799</td>
<td>0.0086</td>
<td>0.8695</td>
<td>0.8891</td>
<td>0.0493</td>
<td>0.0067</td>
<td>0.0413</td>
<td>0.0597</td>
</tr>
<tr>
<td>EWHI1-Ind</td>
<td>0.8779</td>
<td>0.0110</td>
<td>0.8613</td>
<td>0.8900</td>
<td>0.0606</td>
<td>0.0132</td>
<td>0.0433</td>
<td>0.0793</td>
</tr>
<tr>
<td>EWHI2-Ind</td>
<td>0.8800</td>
<td>0.0044</td>
<td>0.8751</td>
<td>0.8848</td>
<td>0.0602</td>
<td>0.0100</td>
<td>0.0451</td>
<td>0.0714</td>
</tr>
<tr>
<td>EMMI2-Ind</td>
<td>0.8879</td>
<td>0.0029</td>
<td>0.8848</td>
<td>0.8916</td>
<td>0.0557</td>
<td>0.0056</td>
<td>0.0497</td>
<td>0.0622</td>
</tr>
<tr>
<td>KEI1-Ind</td>
<td>0.8741</td>
<td>0.0065</td>
<td>0.8650</td>
<td>0.8804</td>
<td>0.0816</td>
<td>0.0092</td>
<td>0.0717</td>
<td>0.0925</td>
</tr>
<tr>
<td>EECI1-Ind</td>
<td>0.8747</td>
<td>0.0029</td>
<td>0.8715</td>
<td>0.8774</td>
<td>0.0748</td>
<td>0.0017</td>
<td>0.0723</td>
<td>0.0765</td>
</tr>
<tr>
<td>EECI2-Ind</td>
<td>0.8807</td>
<td>0.0056</td>
<td>0.8721</td>
<td>0.8867</td>
<td>0.0574</td>
<td>0.0085</td>
<td>0.0480</td>
<td>0.0684</td>
</tr>
</tbody>
</table>

\[ \text{I}_H(\mathcal{P}_{20}^Y) \quad \text{I}_{e+}(\mathcal{P}_Y, \mathcal{P}_{20}^Y) \]
Based on Figures 3.29-3.37, this problem appears much more difficult than the MOP2 problem when starting with an initial design of size 10. In the previous MOP2 function, the best methods were sequentially adding all points at or near the true Pareto set, even when starting with a small initial design. In contrast, with the WSNL function, even the best methods are adding points a considerable distance from the true Pareto set, and several clearly dominated points are added when the initial design contains only five inputs per input dimension.

The results in Table 3.5 and Figures 3.29-3.37 yield several conclusions that are consistent with the MOP2 function, but also yield several conclusions that seem to be in conflict with the previous results. Just as for the MOP2 function, for an initial design of size 10 and 10 sequentially added points, the most effective approaches, in terms of the hypervolume indicator, are those based on the maximin fitness function (EMMI-Ind, EMAX1-Ind, EMMI2-Ind, EMMI-Dep, EMAX1-Dep, and EMMI2-Dep) and the hypervolume indicator (EHI1-Ind, EHI1-Dep, and EHI2-Dep). While the maximin fitness based methods are expectedly performing reasonably well in terms of the binary-$\epsilon$ indicator, it is somewhat surprising that many of the hypervolume-based methods perform better than many maximin fitness based methods in terms of this metric. For example, although EMAX1-Ind has the smallest binary-$\epsilon$ indicator on average, EHI1-Ind, EHI2-Ind, and EHI2-Dep outperform all other maximin fitness based approaches. Additionally, ECI2 does not perform nearly as well, relative to the other approaches, as in the MOP2 problem. In the independence case, both EECI1 and EECI2 outperform ECI2 in terms of the hypervolume indicator, although ECI2 is still slightly better on average than EECI1 and EECI2 in the dependence case in terms of hypervolumes. In general, with the exception of PI-Ind and PI-Dep, all scaling
invariant methods are performing relatively well in this problem. Another surprise is
the effectiveness of the weighted hypervolume indicator, as EWHI1-Ind, EWHI2-Ind,
EWHI1-Dep, and EWHI2-Dep all appear much more competitive when compared
to the hypervolume based methods and the maximin fitness based methods. We
should point out that KEI and KEI1, using both the independence and dependence
model, are by far the most ineffective scaling dependent method.

Perhaps the most curious result based on an initial design of size 10 and 10 sequen-
tially added points is the effectiveness of ECI1-Ind and both EECI2 implementations.
Unlike the MOP2 function, where these approaches only added points in unexplored
regions of the input space and, these scaling invariant methods are quite effective
at adding points at or near the true Pareto front and set in the WSNL function. It
is not exactly clear why this approach could perform so well on one problem and
perform so poorly on another. It is interesting to note from Figure 3.32 that when
ECI1-Ind adds points a considerable distance from the Pareto set, it adds them to
regions that are as far away as possible from the original design points. This also
appears to be the case with the EECI2 implementations.

Just as with the MOP2 problem, there does not appear to be any substantial
increase in performance when using a dependence model as opposed to an indepen-
dence model. While some approaches appear slightly better, on average, using a
dependence model (e.g., EHI2-Ind vs. EHI2-Dep), a majority of the approaches ac-
tually perform worse (in terms of both quality indicators), and even those where the
dependence model offers some advantages, the advantages are slight. Just as with
the MOP2 problem, this could be caused by little or no dependence between the
Table 3.5: Summary of quality indicators for five runs of each algorithm for the WSNL problem. The initial design is a 20-point maximin LHD, and 10 points are added via the expected improvement algorithm.

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean</th>
<th>Std Dev</th>
<th>Min</th>
<th>Max</th>
<th>Mean</th>
<th>Std Dev</th>
<th>P_{30}^*</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMMI-Ind</td>
<td>0.8916</td>
<td>0.0008</td>
<td>0.8902</td>
<td>0.8922</td>
<td>0.0306</td>
<td>0.0019</td>
<td>0.0285</td>
<td>0.0320</td>
<td></td>
</tr>
<tr>
<td>KEI-Ind</td>
<td>0.8790</td>
<td>0.0050</td>
<td>0.8740</td>
<td>0.8850</td>
<td>0.0681</td>
<td>0.0095</td>
<td>0.0613</td>
<td>0.0824</td>
<td></td>
</tr>
<tr>
<td>PI-Ind</td>
<td>0.8397</td>
<td>0.0114</td>
<td>0.8286</td>
<td>0.8579</td>
<td>0.0896</td>
<td>0.0088</td>
<td>0.0806</td>
<td>0.0981</td>
<td></td>
</tr>
<tr>
<td>EMAX1-Ind</td>
<td>0.8919</td>
<td>0.0001</td>
<td>0.8917</td>
<td>0.8920</td>
<td>0.0320</td>
<td>0.0001</td>
<td>0.0318</td>
<td>0.0321</td>
<td></td>
</tr>
<tr>
<td>ECI1-Ind</td>
<td>0.8735</td>
<td>0.0072</td>
<td>0.8689</td>
<td>0.8862</td>
<td>0.0549</td>
<td>0.0060</td>
<td>0.0517</td>
<td>0.0657</td>
<td></td>
</tr>
<tr>
<td>ECI2-Ind</td>
<td>0.8764</td>
<td>0.0072</td>
<td>0.8722</td>
<td>0.8892</td>
<td>0.0511</td>
<td>0.0018</td>
<td>0.0479</td>
<td>0.0524</td>
<td></td>
</tr>
<tr>
<td>EHI1-Ind</td>
<td>0.8899</td>
<td>0.0054</td>
<td>0.8818</td>
<td>0.8942</td>
<td>0.0375</td>
<td>0.0051</td>
<td>0.0301</td>
<td>0.0436</td>
<td></td>
</tr>
<tr>
<td>EHI2-Ind</td>
<td>0.8910</td>
<td>0.0041</td>
<td>0.8861</td>
<td>0.8945</td>
<td>0.0371</td>
<td>0.0067</td>
<td>0.0305</td>
<td>0.0470</td>
<td></td>
</tr>
<tr>
<td>EWHI1-Ind</td>
<td>0.8864</td>
<td>0.0113</td>
<td>0.8663</td>
<td>0.8926</td>
<td>0.0403</td>
<td>0.0116</td>
<td>0.0297</td>
<td>0.0598</td>
<td></td>
</tr>
<tr>
<td>EWHI2-Ind</td>
<td>0.8844</td>
<td>0.0087</td>
<td>0.8735</td>
<td>0.8927</td>
<td>0.0391</td>
<td>0.0048</td>
<td>0.0354</td>
<td>0.0473</td>
<td></td>
</tr>
<tr>
<td>EMMI2-Ind</td>
<td>0.8902</td>
<td>0.0012</td>
<td>0.8892</td>
<td>0.8920</td>
<td>0.0297</td>
<td>0.0005</td>
<td>0.0292</td>
<td>0.0304</td>
<td></td>
</tr>
<tr>
<td>KEI1-Ind</td>
<td>0.8773</td>
<td>0.0059</td>
<td>0.8706</td>
<td>0.8850</td>
<td>0.0739</td>
<td>0.0116</td>
<td>0.0607</td>
<td>0.0824</td>
<td></td>
</tr>
<tr>
<td>EECCI1-Ind</td>
<td>0.8891</td>
<td>0.0010</td>
<td>0.8880</td>
<td>0.8903</td>
<td>0.0396</td>
<td>0.0093</td>
<td>0.0326</td>
<td>0.0502</td>
<td></td>
</tr>
<tr>
<td>EECI2-Ind</td>
<td>0.8879</td>
<td>0.0004</td>
<td>0.8873</td>
<td>0.8883</td>
<td>0.0484</td>
<td>0.0015</td>
<td>0.0463</td>
<td>0.0499</td>
<td></td>
</tr>
<tr>
<td>EMMI-Dep</td>
<td>0.8904</td>
<td>0.0010</td>
<td>0.8894</td>
<td>0.8921</td>
<td>0.0406</td>
<td>0.0052</td>
<td>0.0345</td>
<td>0.0449</td>
<td></td>
</tr>
<tr>
<td>KEI-Dep</td>
<td>0.8816</td>
<td>0.0014</td>
<td>0.8798</td>
<td>0.8835</td>
<td>0.0614</td>
<td>0.0007</td>
<td>0.0606</td>
<td>0.0623</td>
<td></td>
</tr>
<tr>
<td>PI-Dep</td>
<td>0.8515</td>
<td>0.0134</td>
<td>0.8343</td>
<td>0.8662</td>
<td>0.0699</td>
<td>0.0212</td>
<td>0.0429</td>
<td>0.0903</td>
<td></td>
</tr>
<tr>
<td>EMAX1-Dep</td>
<td>0.8901</td>
<td>0.0005</td>
<td>0.8893</td>
<td>0.8904</td>
<td>0.0350</td>
<td>0.0029</td>
<td>0.0300</td>
<td>0.0371</td>
<td></td>
</tr>
<tr>
<td>ECI2-Dep</td>
<td>0.8763</td>
<td>0.0070</td>
<td>0.8721</td>
<td>0.8888</td>
<td>0.0465</td>
<td>0.0063</td>
<td>0.0356</td>
<td>0.0518</td>
<td></td>
</tr>
<tr>
<td>EHI1-Dep</td>
<td>0.8875</td>
<td>0.0050</td>
<td>0.8795</td>
<td>0.8919</td>
<td>0.0402</td>
<td>0.0050</td>
<td>0.0343</td>
<td>0.0457</td>
<td></td>
</tr>
<tr>
<td>EHI2-Dep</td>
<td>0.8883</td>
<td>0.0047</td>
<td>0.8819</td>
<td>0.8943</td>
<td>0.0441</td>
<td>0.0103</td>
<td>0.0313</td>
<td>0.0563</td>
<td></td>
</tr>
<tr>
<td>EWHI1-Dep</td>
<td>0.8834</td>
<td>0.0075</td>
<td>0.8727</td>
<td>0.8907</td>
<td>0.0451</td>
<td>0.0086</td>
<td>0.0380</td>
<td>0.0597</td>
<td></td>
</tr>
<tr>
<td>EWHI2-Dep</td>
<td>0.8841</td>
<td>0.0020</td>
<td>0.8808</td>
<td>0.8863</td>
<td>0.0418</td>
<td>0.0146</td>
<td>0.0309</td>
<td>0.0671</td>
<td></td>
</tr>
<tr>
<td>EMMI2-Dep</td>
<td>0.8916</td>
<td>0.0020</td>
<td>0.8897</td>
<td>0.8939</td>
<td>0.0409</td>
<td>0.0049</td>
<td>0.0344</td>
<td>0.0457</td>
<td></td>
</tr>
<tr>
<td>KEI1-Dep</td>
<td>0.8815</td>
<td>0.0049</td>
<td>0.8774</td>
<td>0.8882</td>
<td>0.0578</td>
<td>0.0107</td>
<td>0.0397</td>
<td>0.0661</td>
<td></td>
</tr>
<tr>
<td>EECI1-Dep</td>
<td>0.8867</td>
<td>0.0022</td>
<td>0.8829</td>
<td>0.8883</td>
<td>0.0495</td>
<td>0.0089</td>
<td>0.0350</td>
<td>0.0568</td>
<td></td>
</tr>
<tr>
<td>EECI2-Dep</td>
<td>0.8896</td>
<td>0.0042</td>
<td>0.8864</td>
<td>0.8966</td>
<td>0.0428</td>
<td>0.0082</td>
<td>0.0332</td>
<td>0.0505</td>
<td></td>
</tr>
</tbody>
</table>
objective functions, a lack of fit of the nonseparable LMC model, or poor estimation of the covariance parameters.

Figure 3.39-3.48 and Table 3.5 both show how important the initial design size can be to the effectiveness of an expected improvement algorithm. The quality indicators are substantially better, and the plots of the estimated Pareto sets and front are much closer to the true Pareto objectives. Other than an overall increase in quality, many of the results based on an initial design of size 20 and 10 sequentially added points lead to the same conclusion as the smaller initial design. The maximin fitness based methods, hypervolume and weighted hypervolume based methods are performing the best, methods based on $I_K(\cdot)$ are performing poorly, and, among the scaling invariant methods, all methods based on the completeness indicator are outperforming the probability of improvement. The estimated completeness indicator methods (EECI and EECI2) are outperforming the averaged completeness indicator approaches (ECI1 and ECI2) in terms of both performance metrics, for both independence and dependence models. The dependence model is offering little or no gain over the independence model when starting with an initial twenty point design.

When the initial design is large enough (i.e., ten inputs per dimension), the plots of the sequential added points, such as Figure 3.39, exhibit just how powerful an expected improvement algorithm can be. Notice how nearly every single sequentially added input is in or extremely close to the true Pareto set. Moreover, notice that each output is on or extremely close to the true Pareto front, and the sequentially added outputs are evenly spread across the true Pareto front.
Figure 3.29: The final run of the five sequential designs for the WSNL problem with an initial design of size 10 and 10 points added sequentially via EMMI-Ind (top), EMAX1-Ind (middle) and EMMI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.30: The final run of the five sequential designs for the WSNL problem with an initial design of size 10 and 10 points added sequentially via EHI1-Ind (top), EWHI1-Ind (middle) and KEI-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.31: The final run of the five sequential designs for the WSNL problem with an initial design of size 10 and 10 points added sequentially via EHI2-Ind (top), EWHI2-Ind (middle) and KEI1-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.32: The final run of the five sequential designs for the WSNL problem with an initial design of size 10 and 10 points added sequentially via PI-Ind (top), ECI1-Ind (middle) and ECI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.33: The final run of the five sequential designs for the WSNL problem with an initial design of size 10 and 10 points added sequentially via EECI1-Ind (top) and EECI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.34: The final run of the five sequential designs for the WSNL problem with an initial design of size 10 and 10 points added sequentially via EMMI-Dep (top), EMAX1-Dep (middle) and EMMI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.35: The final run of the five sequential designs for the WSNL problem with an initial design of size 10 and 10 points added sequentially via EHI1-Dep (top), EWHI1-Dep (middle) and KEI-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.36: The final run of the five sequential designs for the WSNL problem with an initial design of size 10 and 10 points added sequentially via EHI2-Dep (top), EWHI2-Dep (middle) and KEI1-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.37: The final run of the five sequential designs for the WSNL problem with an initial design of size 10 and 10 points added sequentially via PI-Dep (top) and ECI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.38: The final run of the five sequential designs for the WSNL problem with an initial design of size 10 and 10 points added sequentially via EECl1-Dep (top) and EECl2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.39: The final run of the five sequential designs for the WSNL problem with an initial design of size 20 and 10 points added sequentially via EMMI-Ind (top), EMAX1-Ind (middle) and EMMI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.40: The final run of the five sequential designs for the WSNL problem with an initial design of size 20 and 10 points added sequentially via EHI1-Ind (top), EWHI1-Ind (middle) and KEI-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.41: The final run of the five sequential designs for the WSNL problem with an initial design of size 20 and 10 points added sequentially via EHI2-Ind (top), EWHI2-Ind (middle) and KEI1-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.42: The final run of the five sequential designs for the WSNL problem with an initial design of size 20 and 10 points added sequentially via PI-Ind (top), ECI1-Ind (middle) and ECI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.43: The final run of the five sequential designs for the WSNL problem with an initial design of size 20 and 10 points added sequentially via EECI1-Ind (top) and EECI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.44: The final run of the five sequential designs for the WSNL problem with an initial design of size 20 and 10 points added sequentially via EMMI-Dep (top), EMAX1-Dep (middle) and EMMI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.45: The final run of the five sequential designs for the WSNL problem with an initial design of size 20 and 10 points added sequentially via EHI1-Dep (top), EWHI1-Dep (middle) and KEI-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.46: The final run of the five sequential designs for the WSNL problem with an initial design of size 20 and 10 points added sequentially via EHI2-Dep (top), EWHI2-Dep (middle) and KEI1-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.47: The final run of the five sequential designs for the WSNL problem with an initial design of size 20 and 10 points added sequentially via PI-Dep (top) and ECI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.48: The final run of the five sequential designs for the WSNL problem with an initial design of size 20 and 10 points added sequentially via EECI1-Dep (top) and EECI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
This example is designed to have outputs that behave as close as possible to realizations of a Gaussian process model with a nonseparable dependence structure. To do so, we first create a 40 point maximin Latin hypercube $X = (x_1, \ldots, x_{40})^\top$ in $[0, 1]^2$. Then, we simulate 40 bivariate normal random variables

$$Y^{40,2} = Y(x_1), \ldots, Y(x_{40})$$

where each $Y(x_i)$ has mean $[0, 0]^\top$ and

$$\text{cov} (Y(x_i), Y(x_j)) = A \begin{bmatrix} \exp \left\{ \sum_{k=1}^2 \theta_{k,1} (x_{i,k} - x_{j,k})^2 \right\} & 0 \\ 0 & \exp \left\{ -\sum_{k=1}^2 \theta_{k,2} (x_{i,k} - x_{j,k})^2 \right\} \end{bmatrix} A,$$
where $\mathbf{A}$ is the unique, symmetric square-root matrix (defined via eigen decomposition) of

$$
\mathbf{T} = \begin{bmatrix}
1 & -.8 \\
-.8 & 1
\end{bmatrix}
$$

$$
\theta_{1,1} = \theta_{1,2} = 4, \quad \theta_{2,1} = \theta_{2,2} = 5
$$

Therefore, $\mathbf{Y}(\mathbf{x}_1), \ldots, \mathbf{Y}(\mathbf{x}_{40})$ can be seen as realizations of a bivariate Gaussian process centered at the origin with covariance structure specified by (3.1.4) at $\{\mathbf{x}_1, \ldots, \mathbf{x}_{40}\}^\top$.

![Figure 3.51: Dark black circles are the approximated true Pareto front for the DGP function](image1)

![Figure 3.52: Dark black circles are the approximated true Pareto set for the DGP function](image2)

We then fit a dependent Gaussian process model $\mathbf{Y}(\cdot)$ with nonseparable dependence structure based on (3.1.3). The function $DGP$ is then defined as the empirical best linear unbiased predictor of $\mathbf{Y}(\cdot)$, i.e.,

$$
y_{DGP}(\mathbf{x}) = E[\mathbf{Y}(\mathbf{x}) | \mathbf{Y}(\mathbf{x}_1), \ldots, \mathbf{Y}(\mathbf{x}_{40})].
$$
Notice that, for the given realization \( Y(x_1), \ldots, Y(x_{40}) \), \( y_{\text{DGP}}(x) \) is a deterministic function of \( x \). The specific values of \( X \) and \( Y^{40,2} \) used can be found in Table 3.6, and the estimated covariance parameters based on \( X \) and \( Y^{40,2} \) are

\[
\hat{\theta}_{1,1} = 3.8869 \\
\hat{\theta}_{1,2} = 4.1028 \\
\hat{\theta}_{2,1} = 4.6361 \\
\hat{\theta}_{2,2} = 4.3045 \\
\hat{A} = \begin{bmatrix} 0.6896 & -0.4126 \\ -0.4126 & 0.7915 \end{bmatrix}.
\]

Plots of \( y_{\text{DGP}}(x) \) can be seen in Figure (3.49) and Figure (3.50), and the Pareto Front and Set, based on a \( 100 \times 100 \) grid in \([0, 1]^2\) can be see in Figure (3.51) and Figure (3.52).

Since there is no analytic form of the Pareto set or front in this particular problem, a discrete approximation to \( P_X \) and \( P_Y \) was formed by evaluating \( y(x) \) over a \( 100 \times 100 \) grid in the input space. These approximations are plotted in Figure 3.51 and Figure 3.52, and will be used as a basis of comparison for the various methods for approximating the Pareto front.

This example has been constructed as a best-case scenario for the nonseparable LMC, as it is essentially a realization of a bivariate Gaussian process that has a nonseparable LMC covariance structure. If any substantial advantage is available when using the nonseparable dependence model, it should be clearly present in this example function. If little or no gain are seen, in terms of the quality of the estimated Pareto set and front, from using a nonseparable LMC with this function, we should not expect to see huge gains on any multivariate function.
Table 3.6: Elements of $\mathbf{X}$ and $\mathbf{Y}^{40.2}$ used to construct the DGP function.

<table>
<thead>
<tr>
<th></th>
<th>$\mathbf{X}$</th>
<th>$\mathbf{Y}^{40.2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5849</td>
<td>0.3084</td>
<td>1.0057 -1.4698</td>
</tr>
<tr>
<td>0.2008</td>
<td>0.6522</td>
<td>-1.2511 1.5937</td>
</tr>
<tr>
<td>0.4594</td>
<td>0.8144</td>
<td>0.3829 0.3141</td>
</tr>
<tr>
<td>0.1852</td>
<td>0.3498</td>
<td>-1.0782 0.5866</td>
</tr>
<tr>
<td>0.4254</td>
<td>0.3675</td>
<td>0.2069 -0.8975</td>
</tr>
<tr>
<td>0.9824</td>
<td>0.0403</td>
<td>-0.7494 0.2857</td>
</tr>
<tr>
<td>0.1466</td>
<td>0.5805</td>
<td>-1.4306 1.6061</td>
</tr>
<tr>
<td>0.4022</td>
<td>0.5434</td>
<td>-0.3714 0.0240</td>
</tr>
<tr>
<td>0.5643</td>
<td>0.6476</td>
<td>0.4675 -0.3412</td>
</tr>
<tr>
<td>0.1208</td>
<td>0.0114</td>
<td>-1.7120 0.8643</td>
</tr>
<tr>
<td>0.3604</td>
<td>0.4238</td>
<td>-0.3212 -0.2815</td>
</tr>
<tr>
<td>0.8677</td>
<td>0.4819</td>
<td>1.1280 -1.0929</td>
</tr>
<tr>
<td>0.3460</td>
<td>0.7275</td>
<td>-0.4594 0.8880</td>
</tr>
<tr>
<td>0.3143</td>
<td>0.8469</td>
<td>-0.2953 1.0589</td>
</tr>
<tr>
<td>0.7211</td>
<td>0.2918</td>
<td>1.1173 -1.3015</td>
</tr>
<tr>
<td>0.8441</td>
<td>0.0958</td>
<td>0.2018 -0.4753</td>
</tr>
<tr>
<td>0.5366</td>
<td>0.8872</td>
<td>0.7544 0.1250</td>
</tr>
<tr>
<td>0.1541</td>
<td>0.9595</td>
<td>-1.2610 1.8550</td>
</tr>
<tr>
<td>0.2376</td>
<td>0.4467</td>
<td>-0.9908 0.6304</td>
</tr>
<tr>
<td>0.6763</td>
<td>0.4701</td>
<td>0.9681 -1.1630</td>
</tr>
<tr>
<td>0.6158</td>
<td>0.7206</td>
<td>0.7767 -0.2793</td>
</tr>
<tr>
<td>0.8962</td>
<td>0.5704</td>
<td>1.0202 -0.8613</td>
</tr>
<tr>
<td>0.2622</td>
<td>0.2103</td>
<td>-0.4027 -0.2443</td>
</tr>
<tr>
<td>0.9158</td>
<td>0.7713</td>
<td>0.5609 0.0700</td>
</tr>
<tr>
<td>0.7320</td>
<td>0.7838</td>
<td>1.0041 -0.1730</td>
</tr>
<tr>
<td>0.9473</td>
<td>0.9399</td>
<td>0.1746 0.6780</td>
</tr>
<tr>
<td>0.0572</td>
<td>0.3759</td>
<td>-1.6248 1.1790</td>
</tr>
<tr>
<td>0.7660</td>
<td>0.6036</td>
<td>1.0739 -0.8347</td>
</tr>
<tr>
<td>0.0309</td>
<td>0.8570</td>
<td>-1.6392 2.1587</td>
</tr>
<tr>
<td>0.0817</td>
<td>0.2537</td>
<td>-1.5959 0.9147</td>
</tr>
<tr>
<td>0.7901</td>
<td>0.5182</td>
<td>1.1481 -1.0868</td>
</tr>
<tr>
<td>0.6543</td>
<td>0.1754</td>
<td>1.1780 -1.4443</td>
</tr>
<tr>
<td>0.3911</td>
<td>0.1701</td>
<td>0.6021 -1.2097</td>
</tr>
<tr>
<td>0.6309</td>
<td>0.0578</td>
<td>1.1603 -1.4527</td>
</tr>
<tr>
<td>0.2932</td>
<td>0.1264</td>
<td>-0.0853 -0.5382</td>
</tr>
<tr>
<td>0.0051</td>
<td>0.1080</td>
<td>-2.1910 1.2326</td>
</tr>
<tr>
<td>0.4840</td>
<td>0.2392</td>
<td>0.8976 -1.4911</td>
</tr>
<tr>
<td>0.8134</td>
<td>0.9228</td>
<td>0.6636 0.4172</td>
</tr>
<tr>
<td>0.9564</td>
<td>0.6903</td>
<td>0.5626 -0.2381</td>
</tr>
<tr>
<td>0.5132</td>
<td>0.9964</td>
<td>0.4040 0.3939</td>
</tr>
</tbody>
</table>
Table 3.7: Summary of quality indicators for five runs of each algorithm for the DGP problem. The initial design is a 10-point maximin LHD, and 10 points are added via the expected improvement algorithm.

<table>
<thead>
<tr>
<th>Method</th>
<th>( I_H(\mathcal{P}_{30}^{20}) )</th>
<th>( I_{\epsilon^+}(\mathcal{P}<em>Y, \mathcal{P}</em>{30}^{20}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Std Dev</td>
</tr>
<tr>
<td>EMMI-Ind</td>
<td>0.5668</td>
<td>0.0013</td>
</tr>
<tr>
<td>KEI-Ind</td>
<td>0.5559</td>
<td>0.0034</td>
</tr>
<tr>
<td>PI-Ind</td>
<td>0.5268</td>
<td>0.0094</td>
</tr>
<tr>
<td>EMAX1-Ind</td>
<td>0.5673</td>
<td>0.0007</td>
</tr>
<tr>
<td>EC11-Ind</td>
<td>0.5009</td>
<td>0.0074</td>
</tr>
<tr>
<td>EC12-Ind</td>
<td>0.5546</td>
<td>0.0112</td>
</tr>
<tr>
<td>EHI1-Ind</td>
<td>0.5630</td>
<td>0.0029</td>
</tr>
<tr>
<td>EHI2-Ind</td>
<td>0.5593</td>
<td>0.0056</td>
</tr>
<tr>
<td>EWH11-Ind</td>
<td>0.5631</td>
<td>0.0049</td>
</tr>
<tr>
<td>EWH21-Ind</td>
<td>0.5557</td>
<td>0.0046</td>
</tr>
<tr>
<td>EMM21-Ind</td>
<td>0.5680</td>
<td>0.0013</td>
</tr>
<tr>
<td>KE11-Ind</td>
<td>0.5542</td>
<td>0.0035</td>
</tr>
<tr>
<td>EEC11-Ind</td>
<td>0.5558</td>
<td>0.0106</td>
</tr>
<tr>
<td>EEC12-Ind</td>
<td>0.5486</td>
<td>0.0120</td>
</tr>
<tr>
<td>EMMI-Dep</td>
<td>0.5676</td>
<td>0.0003</td>
</tr>
<tr>
<td>KEI-Dep</td>
<td>0.5473</td>
<td>0.0072</td>
</tr>
<tr>
<td>PI-Dep</td>
<td>0.5242</td>
<td>0.0167</td>
</tr>
<tr>
<td>EMAX1-Dep</td>
<td>0.5669</td>
<td>0.0006</td>
</tr>
<tr>
<td>ECI2-Dep</td>
<td>0.5526</td>
<td>0.0120</td>
</tr>
<tr>
<td>EHI1-Dep</td>
<td>0.5620</td>
<td>0.0027</td>
</tr>
<tr>
<td>EHI2-Dep</td>
<td>0.5622</td>
<td>0.0019</td>
</tr>
<tr>
<td>EWH11-Dep</td>
<td>0.5599</td>
<td>0.0033</td>
</tr>
<tr>
<td>EWH21-Dep</td>
<td>0.5583</td>
<td>0.0011</td>
</tr>
<tr>
<td>EMM21-Dep</td>
<td>0.5660</td>
<td>0.0031</td>
</tr>
<tr>
<td>KE11-Dep</td>
<td>0.5556</td>
<td>0.0046</td>
</tr>
<tr>
<td>EEC11-Dep</td>
<td>0.5605</td>
<td>0.0023</td>
</tr>
<tr>
<td>EEC12-Dep</td>
<td>0.5394</td>
<td>0.0115</td>
</tr>
</tbody>
</table>
The results in Table 3.7 and Table 3.8 do not show a considerable advantage in using the nonseparable LMC model. In Table 3.7, only five methods (EMMI, EHI2, EWHI2, KEI1, and EECI1) have better quality indicators, on average, for the dependence model. And, among the four that see an increase in quality, these increases are slight. Moreover, if we examine Figures 3.53-3.62, visual differences in the quality of the estimated Pareto fronts and sets are barely noticeable.

Table 3.8 shows that a majority of expected improvement algorithms show no increase, on average, in the hypervolume indicator. However, all methods, other than EHI2, PI, EECI1, EECI2, and EMMI, show a decrease in the binary-$\epsilon$ indicator. Again, most of these increases in performance are slight, and are not clearly manifested in Figures 3.63-3.72.

The nonseparable LMC is completely without merit; it produces reasonable approximations of the Pareto front. However, it appears to have little advantage over the computationally simpler independence model when approximating the Pareto front. It appears that the steep computational burden of the nonseparable LMC is not offset by tremendous gains in the quality of the approximation of the Pareto front and set.

In addition to the lack of clear superiority of the dependence model, many of these results are consistent with the previous examples. For example, maximin fitness based methods and hypervolume based methods appear to be most effective. Weighted hypervolume based methods appear more competitive than some previous examples. Additionally, ECI1 is largely ineffective, in contrast to the WSNL example. It appears to over-explore the input space and favor points outside of the convex hull of the initial design, which is consistent with the MOP2 example. Somewhat surprisingly, EECI2 is reasonably effective, which is in sharp contrast to the MOP2 results. Three of the four
Table 3.8: Summary of quality indicators in five runs of each algorithm for the \textit{DGP} problem. The initial design is a 20-point maximin LHD, and 10 points are added via the expected improvement algorithm.

<table>
<thead>
<tr>
<th>Method</th>
<th>(I_H(\mathcal{P}_{30}^0))</th>
<th>(I_{e_+}(\mathcal{P}<em>Y, \mathcal{P}</em>{30}^0))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Std Dev</td>
</tr>
<tr>
<td>EMMI-Ind</td>
<td>0.5732</td>
<td>0.0004</td>
</tr>
<tr>
<td>KEI-Ind</td>
<td>0.5621</td>
<td>0.0031</td>
</tr>
<tr>
<td>PI-Ind</td>
<td>0.5261</td>
<td>0.0207</td>
</tr>
<tr>
<td>EMAX1-Ind</td>
<td>0.5731</td>
<td>0.0005</td>
</tr>
<tr>
<td>ECI1-Ind</td>
<td>0.5246</td>
<td>0.0043</td>
</tr>
<tr>
<td>ECI2-Ind</td>
<td>0.5444</td>
<td>0.0027</td>
</tr>
<tr>
<td>EHI1-Ind</td>
<td>0.5668</td>
<td>0.0036</td>
</tr>
<tr>
<td>EHI2-Ind</td>
<td>0.5673</td>
<td>0.0022</td>
</tr>
<tr>
<td>EWHI1-Ind</td>
<td>0.5679</td>
<td>0.0021</td>
</tr>
<tr>
<td>EWHI2-Ind</td>
<td>0.5624</td>
<td>0.0043</td>
</tr>
<tr>
<td>EMMI2-Ind</td>
<td>0.5695</td>
<td>0.0023</td>
</tr>
<tr>
<td>KEI1-Ind</td>
<td>0.5610</td>
<td>0.0017</td>
</tr>
<tr>
<td>EECI1-Ind</td>
<td>0.5399</td>
<td>0.0014</td>
</tr>
<tr>
<td>EECI2-Ind</td>
<td>0.5534</td>
<td>0.0036</td>
</tr>
<tr>
<td>EMMI-Dep</td>
<td>0.5720</td>
<td>0.0014</td>
</tr>
<tr>
<td>KEI-Dep</td>
<td>0.5612</td>
<td>0.0060</td>
</tr>
<tr>
<td>PI-Dep</td>
<td>0.5164</td>
<td>0.0122</td>
</tr>
<tr>
<td>EMAX1-Dep</td>
<td>0.5725</td>
<td>0.0008</td>
</tr>
<tr>
<td>ECI2-Dep</td>
<td>0.5428</td>
<td>0.0026</td>
</tr>
<tr>
<td>EHI1-Dep</td>
<td>0.5694</td>
<td>0.0012</td>
</tr>
<tr>
<td>EHI2-Dep</td>
<td>0.5663</td>
<td>0.0039</td>
</tr>
<tr>
<td>EWHI1-Dep</td>
<td>0.5655</td>
<td>0.0053</td>
</tr>
<tr>
<td>EWHI2-Dep</td>
<td>0.5657</td>
<td>0.0021</td>
</tr>
<tr>
<td>EMMI2-Dep</td>
<td>0.5716</td>
<td>0.0009</td>
</tr>
<tr>
<td>KEI1-Dep</td>
<td>0.5602</td>
<td>0.0049</td>
</tr>
<tr>
<td>EECI1-Dep'</td>
<td>0.5398</td>
<td>0.0019</td>
</tr>
<tr>
<td>EECI2-Dep'</td>
<td>0.5428</td>
<td>0.0047</td>
</tr>
</tbody>
</table>
scaling invariant methods based on the completeness indicator (EECI1, EECI2, and
ECI2) produced relatively high-quality approximations. Expectedly, higher quality
Pareto fronts are constructed when using a 20 point initial design instead of a 10
point initial design.
Figure 3.53: The final run of the five sequential designs for the DGP problem with an initial design of size 10 and 10 points added sequentially via EMMI-Ind (top), EMAX1-Ind (middle) and EMMI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.54: The final run of the five sequential designs for the DGP problem with an initial design of size 10 and 10 points added sequentially via EHI1-Ind (top), EWHI1-Ind (middle) and KEI-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.55: The final run of the five sequential designs for the DGP problem with an initial design of size 10 and 10 points added sequentially via EHI2-Ind (top), EWHI2-Ind (middle) and KEI1-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.56: The final run of the five sequential designs for the DGP problem with an initial design of size 10 and 10 points added sequentially via PI-Ind (top), ECI1-Ind (middle) and ECI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.57: The final run of the five sequential designs for the DGP problem with an initial design of size 10 and 10 points added sequentially via EECI1-Ind (top) and EECI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.58: The final run of the five sequential designs for the DGP problem with an initial design of size 10 and 10 points added sequentially via EMMI-Dep (top), EMAX1-Dep (middle) and EMMI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.59: The final run of the five sequential designs for the DGP problem with an initial design of size 10 and 10 points added sequentially via EHII1-Dep (top), EWHII1-Dep (middle) and KEI-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.60: The final run of the five sequential designs for the DGP problem with an initial design of size 10 and 10 points added sequentially via EHI2-Dep (top), EWHI2-Dep (middle) and KEI1-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.61: The final run of the five sequential designs for the DGP problem with an initial design of size 10 and 10 points added sequentially via PI-Dep (top) and ECI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.62: The final run of the five sequential designs for the DGP problem with an initial design of size 10 and 10 points added sequentially via EECI1-Dep (top) and EECI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.63: The final run of the five sequential designs for the DGP problem with an initial design of size 20 and 10 points added sequentially via EMMI-Ind (top), EMAX1-Ind (middle) and EMMI2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.64: The final run of the five sequential designs for the DGP problem with an initial design of size 20 and 10 points added sequentially via EHI1-Ind (top), EWHI1-Ind (middle) and KEI-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.65: The final run of the five sequential designs for the DGP problem with an initial design of size 20 and 10 points added sequentially via EHI2-Ind (top), EWHI2-Ind (middle) and KEI1-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.66: The final run of the five sequential designs for the DGP problem with an initial design of size 20 and 10 points added sequentially via PI-Ind (top), ECII-Ind (middle) and ECII2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.67: The final run of the five sequential designs for the DGP problem with an initial design of size 20 and 10 points added sequentially via EECL1-Ind (top) and EECL2-Ind (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.68: The final run of the five sequential designs for the DGP problem with an initial design of size 20 and 10 points added sequentially via EMMI-Dep (top), EMAX1-Dep (middle) and EMMI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.69: The final run of the five sequential designs for the DGP problem with an initial design of size 20 and 10 points added sequentially via EHI1-Dep (top), EWHI1-Dep (middle) and KEI-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.70: The final run of the five sequential designs for the DGP problem with an initial design of size 20 and 10 points added sequentially via EHI2-Dep (top), EWHI2-Dep (middle) and KEI1-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.71: The final run of the five sequential designs for the DGP problem with an initial design of size 20 and 10 points added sequentially via PI-Dep (top) and ECI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
Figure 3.72: The final run of the five sequential designs for the DGP problem with an initial design of size 20 and 10 points added sequentially via EECII-Dep (top) and EECI2-Dep (bottom). The lighter green region is the true Pareto set and front, the magenta circles are the initial design, and the black crosses are the sequentially added points.
3.1.4 DTLZ2

This example evaluates the performance of the various methods in a higher-dimensional case. To do so, the DTLZ2 test function, described in Deb et al. (2005), is used. DTLZ2 was designed to be scalable in both the number of inputs and outputs. This example considers the case where there are $m = 4$ outputs and $d = 4$ inputs. The input space is $\mathcal{X} = [0, 1]^4$. The outputs are

\[
y_1(x) = (1 + g(x_4)) \cos\left(\frac{\pi x_1}{2}\right) \cos\left(\frac{\pi x_2}{2}\right) \cos\left(\frac{\pi x_3}{2}\right) \tag{3.1.4}
\]

\[
y_2(x) = (1 + g(x_4)) \sin\left(\frac{\pi x_3}{2}\right) \cos\left(\frac{\pi x_1}{2}\right) \cos\left(\frac{\pi x_2}{2}\right) \tag{3.1.5}
\]

\[
y_3(x) = (1 + g(x_4)) \sin\left(\frac{\pi x_2}{2}\right) \cos\left(\frac{\pi x_1}{2}\right) \tag{3.1.6}
\]

\[
y_4(x) = (1 + g(x_4)) \sin\left(\frac{\pi x_1}{2}\right) \tag{3.1.7}
\]

where

\[
g(x_4) = (x_4 - 0.5)^2. \tag{3.1.8}
\]

The Pareto set is $\mathcal{P}_X = \{x : x_4 = 0.5\}$ and $\mathcal{P}_Y$ is the concave set where $g(x_4) = 0$. A discrete approximation to $\mathcal{P}_Y$ was created by evaluating DTLZ2 at 20,000 points uniformly spread in $\mathcal{P}_X$.

Because of the output space has four dimensions instead of two, there are some additional difficulties in this problem. First, some methods (KEI1-Ind, ECI1-Ind, KEI2-Ind, EWHI1-Dep) have not been tested at all, as their implementations require exceedingly long computation times. Additionally, when an initial design of 40 points (10 per dimension) is used, only a limited number of improvement criteria were used, since five runs of the expected improvement algorithms can require excessively long run times. Specifically, in addition to the previously mentioned untested improvement
Table 3.9: Summary of quality indicators in five runs of each algorithm for the DTLZ2 problem. The initial design is a 20-point maximin LHD, and 20 points are added via the expected improvement algorithm.

<table>
<thead>
<tr>
<th>Method</th>
<th>$I_H(P^0_Y)$ Mean</th>
<th>Std Dev</th>
<th>Min</th>
<th>Max</th>
<th>$I_{e^+}(P_Y, P^0_Y)$ Mean</th>
<th>Std Dev</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMMI-Ind</td>
<td>0.7380</td>
<td>0.0061</td>
<td>0.7295</td>
<td>0.7444</td>
<td>0.2430</td>
<td>0.0081</td>
<td>0.2368</td>
<td>0.2569</td>
</tr>
<tr>
<td>KEI-Ind</td>
<td>0.6689</td>
<td>0.0292</td>
<td>0.6323</td>
<td>0.7120</td>
<td>0.3023</td>
<td>0.0317</td>
<td>0.2557</td>
<td>0.3324</td>
</tr>
<tr>
<td>PI-Ind</td>
<td>0.5968</td>
<td>0.0215</td>
<td>0.5738</td>
<td>0.6261</td>
<td>0.4291</td>
<td>0.0345</td>
<td>0.3675</td>
<td>0.4476</td>
</tr>
<tr>
<td>EMAX1-Ind</td>
<td>0.7300</td>
<td>0.0035</td>
<td>0.7265</td>
<td>0.7344</td>
<td>0.2402</td>
<td>0.0040</td>
<td>0.2347</td>
<td>0.2442</td>
</tr>
<tr>
<td>ECI2-Ind</td>
<td>0.6360</td>
<td>0.0386</td>
<td>0.5954</td>
<td>0.6845</td>
<td>0.3917</td>
<td>0.0554</td>
<td>0.3190</td>
<td>0.4476</td>
</tr>
<tr>
<td>EHI1-Ind</td>
<td>0.7117</td>
<td>0.0265</td>
<td>0.6785</td>
<td>0.7369</td>
<td>0.3818</td>
<td>0.0320</td>
<td>0.3675</td>
<td>0.4390</td>
</tr>
<tr>
<td>EHI2-Ind</td>
<td>0.7183</td>
<td>0.0213</td>
<td>0.6921</td>
<td>0.7454</td>
<td>0.3626</td>
<td>0.0778</td>
<td>0.2684</td>
<td>0.4435</td>
</tr>
<tr>
<td>EWHI1-Ind</td>
<td>0.6940</td>
<td>0.0338</td>
<td>0.6510</td>
<td>0.7273</td>
<td>0.3547</td>
<td>0.0614</td>
<td>0.2986</td>
<td>0.4476</td>
</tr>
<tr>
<td>EWHI2-Ind</td>
<td>0.7035</td>
<td>0.0117</td>
<td>0.6884</td>
<td>0.7206</td>
<td>0.3390</td>
<td>0.0410</td>
<td>0.2927</td>
<td>0.3718</td>
</tr>
<tr>
<td>EMMI2-Ind</td>
<td>0.7361</td>
<td>0.0037</td>
<td>0.7314</td>
<td>0.7409</td>
<td>0.2393</td>
<td>0.0049</td>
<td>0.2348</td>
<td>0.2476</td>
</tr>
<tr>
<td>EECCI1-Ind</td>
<td>0.6188</td>
<td>0.0268</td>
<td>0.5954</td>
<td>0.6619</td>
<td>0.4304</td>
<td>0.0327</td>
<td>0.3726</td>
<td>0.4476</td>
</tr>
<tr>
<td>EECCI2-Ind</td>
<td>0.5238</td>
<td>0.0028</td>
<td>0.5194</td>
<td>0.5265</td>
<td>0.4476</td>
<td>0.0000</td>
<td>0.4476</td>
<td>0.4476</td>
</tr>
<tr>
<td>EMMI-Dep</td>
<td>0.6959</td>
<td>0.0167</td>
<td>0.6694</td>
<td>0.7129</td>
<td>0.3020</td>
<td>0.0075</td>
<td>0.2939</td>
<td>0.3109</td>
</tr>
<tr>
<td>KEI-Dep</td>
<td>0.6894</td>
<td>0.0278</td>
<td>0.6445</td>
<td>0.7193</td>
<td>0.2980</td>
<td>0.0178</td>
<td>0.2762</td>
<td>0.3192</td>
</tr>
<tr>
<td>PI-Dep</td>
<td>0.6273</td>
<td>0.0272</td>
<td>0.5887</td>
<td>0.6563</td>
<td>0.3926</td>
<td>0.0681</td>
<td>0.2838</td>
<td>0.4435</td>
</tr>
<tr>
<td>EMAX1-Dep</td>
<td>0.7106</td>
<td>0.0105</td>
<td>0.7001</td>
<td>0.7239</td>
<td>0.2853</td>
<td>0.0094</td>
<td>0.2743</td>
<td>0.2995</td>
</tr>
<tr>
<td>ECI2-Dep</td>
<td>0.6711</td>
<td>0.0282</td>
<td>0.6349</td>
<td>0.6999</td>
<td>0.3363</td>
<td>0.0253</td>
<td>0.3154</td>
<td>0.3675</td>
</tr>
<tr>
<td>EHI1-Dep</td>
<td>0.6803</td>
<td>0.0188</td>
<td>0.6621</td>
<td>0.7119</td>
<td>0.3446</td>
<td>0.0271</td>
<td>0.2995</td>
<td>0.3718</td>
</tr>
<tr>
<td>EHI2-Dep</td>
<td>0.7022</td>
<td>0.0173</td>
<td>0.6726</td>
<td>0.7149</td>
<td>0.3200</td>
<td>0.0300</td>
<td>0.2981</td>
<td>0.3675</td>
</tr>
<tr>
<td>EWHI2-Dep</td>
<td>0.6742</td>
<td>0.0283</td>
<td>0.6281</td>
<td>0.7011</td>
<td>0.3337</td>
<td>0.0320</td>
<td>0.2981</td>
<td>0.3686</td>
</tr>
<tr>
<td>EMMI2-Dep</td>
<td>0.6761</td>
<td>0.0252</td>
<td>0.6537</td>
<td>0.7174</td>
<td>0.3242</td>
<td>0.0387</td>
<td>0.2707</td>
<td>0.3634</td>
</tr>
<tr>
<td>EECCI1-Dep</td>
<td>0.6935</td>
<td>0.0187</td>
<td>0.6724</td>
<td>0.7225</td>
<td>0.3142</td>
<td>0.0261</td>
<td>0.2860</td>
<td>0.3560</td>
</tr>
<tr>
<td>EECCI2-Dep</td>
<td>0.5537</td>
<td>0.0352</td>
<td>0.5240</td>
<td>0.5992</td>
<td>0.4326</td>
<td>0.0335</td>
<td>0.3726</td>
<td>0.4476</td>
</tr>
</tbody>
</table>
criteria, EMAX1-Dep, EMMI2-Dep, and EWHI2-Dep were not used on the 40 point initial design. Additionally, since there is not a straightforward way to visualize the estimated Pareto front and set, we will rely solely on the hypervolume indicator and binary-$\epsilon$ indicator to evaluate the quality of the various improvement criteria.

Based on the 20-point initial design (5 inputs per dimension) results in Table 3.9, there are several trends that are consistent with the two-dimensional results. First, the maximin fitness based methods and hypervolume based methods with independence models (EMMI-Ind, EMAX1-Ind, EMMI2-Ind, EHI1-Ind,EHI2-Ind) seem to perform the best in terms of the hypervolume indicator. Both hypervolume-based methods with independence models have hypervolumes, on average, greater than 0.71, and the maximin fitness-based have average hypervolumes greater than 0.73. Also, judging by average hypervolumes, the other scaling-dependent approaches (KEI and the methods based on the weighted hypervolume indicator) are being consistently outperformed by methods based on the hypervolume indicator and maximin fitness function.

However, this higher-dimensional example also yields some different results. For example, among all scaling invariant methods tested (ECI2, PI, EECI1, EECI2), the dependence models appeared to perform substantially better, in terms of both the hypervolume indicator and the binary-$\epsilon$ indicator, than the independence models. Additionally, more so than any other examples, the two quality indicators used seem to be in conflict with each other. For example, KEI, for both the independence and dependence models, appears to be below average among scaling dependent method. However, in terms of the binary-$\epsilon$ indicator, it appears to be better than all scaling dependent methods that are not based on the maximin fitness function. Similarly, the hypervolume based methods EHI1 and EHI2 perform quite poorly in terms of the...
Table 3.10: Summary of quality indicators in five runs of each algorithm for the DTLZ2 problem. The initial design is a 40-point maximin LHD, and 20 points are added via the expected improvement algorithm.

<table>
<thead>
<tr>
<th>Method</th>
<th>(I_H(P_{60}^{60})) Mean</th>
<th>(I_H(P_{60}^{60})) Std Dev</th>
<th>(I_H(P_{60}^{60})) Min</th>
<th>(I_H(P_{60}^{60})) Max</th>
<th>(I_+ (P_Y, P_{60}^{60})) Mean</th>
<th>(I_+ (P_Y, P_{60}^{60})) Std Dev</th>
<th>(I_+ (P_Y, P_{60}^{60})) Min</th>
<th>(I_+ (P_Y, P_{60}^{60})) Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMMI-Ind</td>
<td>0.7505</td>
<td>0.0020</td>
<td>0.7472</td>
<td>0.7522</td>
<td>0.2056</td>
<td>0.0033</td>
<td>0.1998</td>
<td>0.2081</td>
</tr>
<tr>
<td>KEI-Ind</td>
<td>0.6771</td>
<td>0.0032</td>
<td>0.6731</td>
<td>0.6819</td>
<td>0.2964</td>
<td>0.0366</td>
<td>0.2685</td>
<td>0.3364</td>
</tr>
<tr>
<td>PI-Ind</td>
<td>0.6531</td>
<td>0.0147</td>
<td>0.6305</td>
<td>0.6668</td>
<td>0.2911</td>
<td>0.0137</td>
<td>0.2762</td>
<td>0.3057</td>
</tr>
<tr>
<td>EMAX1-Ind</td>
<td>0.7485</td>
<td>0.0032</td>
<td>0.7430</td>
<td>0.7512</td>
<td>0.2076</td>
<td>0.0151</td>
<td>0.1915</td>
<td>0.2318</td>
</tr>
<tr>
<td>ECI2-Ind</td>
<td>0.6648</td>
<td>0.0087</td>
<td>0.6534</td>
<td>0.6778</td>
<td>0.3063</td>
<td>0.0095</td>
<td>0.3020</td>
<td>0.3234</td>
</tr>
<tr>
<td>EHI1-Ind</td>
<td>0.7457</td>
<td>0.0048</td>
<td>0.7389</td>
<td>0.7520</td>
<td>0.2559</td>
<td>0.0320</td>
<td>0.1987</td>
<td>0.2702</td>
</tr>
<tr>
<td>EHI2-Ind</td>
<td>0.7539</td>
<td>0.0059</td>
<td>0.7487</td>
<td>0.7638</td>
<td>0.2663</td>
<td>0.0088</td>
<td>0.2505</td>
<td>0.2702</td>
</tr>
<tr>
<td>EWH1-Ind</td>
<td>0.7277</td>
<td>0.0132</td>
<td>0.7127</td>
<td>0.7393</td>
<td>0.2709</td>
<td>0.0151</td>
<td>0.2503</td>
<td>0.2867</td>
</tr>
<tr>
<td>EWH2-Ind</td>
<td>0.7325</td>
<td>0.0103</td>
<td>0.7157</td>
<td>0.7428</td>
<td>0.2642</td>
<td>0.0125</td>
<td>0.2485</td>
<td>0.2798</td>
</tr>
<tr>
<td>EMMI2-Ind</td>
<td>0.7548</td>
<td>0.0027</td>
<td>0.7523</td>
<td>0.7590</td>
<td>0.2054</td>
<td>0.0036</td>
<td>0.2017</td>
<td>0.2100</td>
</tr>
<tr>
<td>EECI1-Ind</td>
<td>0.6736</td>
<td>0.0297</td>
<td>0.6272</td>
<td>0.7245</td>
<td>0.3147</td>
<td>0.0119</td>
<td>0.3013</td>
<td>0.3234</td>
</tr>
<tr>
<td>EECI2-Ind</td>
<td>0.6109</td>
<td>0.0023</td>
<td>0.6068</td>
<td>0.6123</td>
<td>0.3364</td>
<td>0.0000</td>
<td>0.3364</td>
<td>0.3364</td>
</tr>
<tr>
<td>EMMI-Dep</td>
<td>0.7548</td>
<td>0.0027</td>
<td>0.7523</td>
<td>0.7590</td>
<td>0.2054</td>
<td>0.0036</td>
<td>0.2017</td>
<td>0.2100</td>
</tr>
<tr>
<td>KEI-Dep</td>
<td>0.6888</td>
<td>0.0178</td>
<td>0.6642</td>
<td>0.7137</td>
<td>0.2993</td>
<td>0.0127</td>
<td>0.2901</td>
<td>0.3214</td>
</tr>
<tr>
<td>PI-Dep</td>
<td>0.6772</td>
<td>0.0208</td>
<td>0.6495</td>
<td>0.7054</td>
<td>0.2793</td>
<td>0.0196</td>
<td>0.2521</td>
<td>0.3013</td>
</tr>
<tr>
<td>ECI2-Dep</td>
<td>0.6528</td>
<td>0.0149</td>
<td>0.6379</td>
<td>0.6779</td>
<td>0.3161</td>
<td>0.0163</td>
<td>0.2871</td>
<td>0.3234</td>
</tr>
<tr>
<td>EHI1-Dep</td>
<td>0.7036</td>
<td>0.0397</td>
<td>0.6430</td>
<td>0.7407</td>
<td>0.2925</td>
<td>0.0268</td>
<td>0.2702</td>
<td>0.3234</td>
</tr>
<tr>
<td>EHI2-Dep</td>
<td>0.6759</td>
<td>0.0425</td>
<td>0.6196</td>
<td>0.7284</td>
<td>0.3063</td>
<td>0.0291</td>
<td>0.2726</td>
<td>0.3364</td>
</tr>
<tr>
<td>EECI1-Dep</td>
<td>0.6658</td>
<td>0.0420</td>
<td>0.6137</td>
<td>0.7290</td>
<td>0.3126</td>
<td>0.0255</td>
<td>0.2702</td>
<td>0.3364</td>
</tr>
<tr>
<td>EECI2-Dep</td>
<td>0.6138</td>
<td>0.0074</td>
<td>0.6083</td>
<td>0.6265</td>
<td>0.3364</td>
<td>0.0000</td>
<td>0.3364</td>
<td>0.3364</td>
</tr>
</tbody>
</table>

binary-\(\epsilon\) indicator, which is in stark contrast to their rather high average hypervolume values.

Here are some other observations from Table 3.9. Among all scaling invariant methods, EECI1-Dep is the best, on average, in terms of both the binary-\(\epsilon\) indicator and the hypervolume indicator. ECI2-Ind is the best scaling invariant method, in terms of both quality measures, when assuming independence. Also, with an independence model, the scaling invariant methods all perform noticeably worse than all the

198
scaling dependence methods. However, with a dependence model, this performance gap is not nearly as big, and some scaling invariant methods, specifically as ECI2 and EECI1, appear to be just as good, if not better, than many of the scaling dependent approaches.

When starting with an initial design of size 40 (10 inputs per dimension), there are some perplexing results. Most surprisingly, when comparing the results with the dependence model in Table 3.10 with those in Table 3.9, it appears that in several cases, the approximation created with the 20 point design is outperforming the approximation with the 40 point design in terms of the average hypervolume indicator. For example, in Table 3.10, ECI2-Dep has an average hypervolume of 0.6528, compared an average hypervolume of 0.6711 in Table 3.9. This only appears to occur with the dependence model, as the approximation created with an independence model appear to have better performance with the larger initial design. Perhaps these results can be attributed to the difficulty in fitting the dependence model, which requires optimization of a restricted likelihood function of 26 unknown parameters.

Other than this anomaly, however, the most successful approaches when starting with a 20-point initial design appear to be the best in this 40-point initial design setting. The maximin fitness based methods and the hypervolume based methods the overall best approaches in terms of both the hypervolume indicator and the binary-ε indicator. EMMI, EHI1, and EHI2 appear worse when using the dependence model. In regard to the scale invariant methods, EECI1-Ind and ECI2-Ind appear to be the most effective methods, but in contrast to the 20-point initial design, their implementations with the dependence model are now slightly worse.
3.1.5 ZLT1

Like the DTLZ2 example, the ZLT1 example is presented to evaluate the performance of the various methods in higher-dimensional cases. This function, described in Laumanns (2005) and sometimes referred to as the “sphere model”, contains $d = 6$ inputs and $m = 6$ outputs. The input space is $\mathcal{X} = [0,1]^6$, and the outputs are defined as

$$y_i(x) = 1 + 0.5(x_i - 1)^2 + \sum_{j=1, j \neq i}^{m} x_j^2, \quad i = 1, \ldots, 6. \quad (3.1.9)$$

The Pareto set is $\mathcal{P}_\mathcal{X} = \{x : x_1 + \ldots + x_6 = 1\}$, and the corresponding Pareto front is convex.

Because the dimension of the input and output space are considerably higher than other examples, five runs of each expected improvement algorithm is both computationally demanding and time-consuming in the independence case. In the dependence case, these difficulties are further magnified. Therefore, for this particular test function, we performed one run of each expected improvement algorithm, using only the independence model. Moreover, just as in the DTLZ2 example, we will not use ECI1 and KEI1, as these methods require excessively long computations.

Based on only this single run for each output, we see that the results for our highest dimensional example are mostly consistent with the lower-dimensional examples. Among scale dependent methods, the hypervolume-based approaches (EHI1-Ind and EHI2-Ind) perform the best in terms of both the hypervolume indicator. Not far behind are the maximin fitness based methods (EMMI-Ind, EMMI2-Ind, and EMAX-Ind). The scale invariant ECI2 performs the best in terms of the binary-$\epsilon$ indicator. Surprisingly, the hypervolume based methods outperform the three maximin fitness
Table 3.11: Summary of quality indicators for the ZLT1 problem. Algorithms were run only one time for each improvement criteria, using only independence models for the output.

<table>
<thead>
<tr>
<th>Method</th>
<th>$I_H(\mathcal{P}_Y^{60})$</th>
<th>$I_{+}(\mathcal{P}_Y, \mathcal{P}_Y^{60})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMMI-Ind</td>
<td>0.6373</td>
<td>0.0893</td>
</tr>
<tr>
<td>KEI-Ind</td>
<td>0.4620</td>
<td>0.1693</td>
</tr>
<tr>
<td>PI-Ind</td>
<td>0.5254</td>
<td>0.1219</td>
</tr>
<tr>
<td>EMAX1-Ind</td>
<td>0.6431</td>
<td>0.0944</td>
</tr>
<tr>
<td>ECI2-Ind</td>
<td>0.6543</td>
<td>0.0606</td>
</tr>
<tr>
<td>EHI1-Ind</td>
<td>0.6680</td>
<td>0.0723</td>
</tr>
<tr>
<td>EHI2-Ind</td>
<td>0.6641</td>
<td>0.0650</td>
</tr>
<tr>
<td>EWHI1-Ind</td>
<td>0.6156</td>
<td>0.0960</td>
</tr>
<tr>
<td>EWHI2-Ind</td>
<td>0.5807</td>
<td>0.0883</td>
</tr>
<tr>
<td>EMMI2-Ind</td>
<td>0.6439</td>
<td>0.0924</td>
</tr>
<tr>
<td>EECI1-Ind</td>
<td>0.5917</td>
<td>0.0734</td>
</tr>
<tr>
<td>EECI2-Ind</td>
<td>0.4107</td>
<td>0.1851</td>
</tr>
</tbody>
</table>

Based methods, which is counter-intuitive in the case of EMMI when one considers its close relationship to the binary-$\epsilon$ indicator. Consistent with previous results, the weighted hypervolume based approaches (EWHI1-Ind and EWHI2-Ind) and KEI-Ind are the worst scale dependent methods. Among the scale invariant methods, EECI1-Ind and ECI2-Ind appear to be the top performers, while PI-Ind and EECI2-Ind produce rather poor Pareto front approximations.

Although plots of the estimated Pareto front and set cannot be made in this higher-dimensional input/output setting, we can still use graphically assess how close the Pareto set approximation is to the true Pareto set. Recall that in the ZLT1 function, the true Pareto set consists of all inputs such that the sum of the components equal to one. Therefore, we can calculate the sum of the components of the inputs in each Pareto set approximation, and we can then make boxplots of these sums. A
Figure 3.73: Boxplots of the component sums for each input in the Pareto set approximation created by the expected improvement criteria indicated on the horizontal axis for the ZLT1 problem. Component sums close to 1 are indicative of inputs near the true Pareto set.
boxplot that is centered close to 1 with very little spread would indicate a good approximation, while sums that considerably deviate from 1, or a boxplot with a center that is considerable distance from 1, would indicate a poor approximation.

The boxplots support most of the conclusions made from the table. In particular, KEI, PI, and EECI2 all perform poorly, as it appears that very few of the elements in the estimated Pareto set created using these criteria have components that sum to 1. Additionally, EMMI, EMAX, EMMI1, EHI1, EHI2, and ECI2 all have boxplots that are centered at 1. So, the plot indicates that these methods perform well, which is in agreement with conclusion drawn from Table 3.11.

However, the Pareto set quality indicators and Figure 3.73 are not in total agreement. Judging by the boxplots, the maximin fitness based methods appear to be creating better approximations, as there appear to be fewer inputs in the Pareto set approximations with sums that are considerably different from 1. Notice that EMMI, EMMI2, and EMAX1 all have boxplots that have much less deviation from 1. So, judging solely by the boxplot, one would conclude that these methods are superior to the hypervolume based methods. However, the results in the boxplot should be taken with caution, as these boxplots do not take into account the amount of spread in the Pareto set or Pareto front approximation.

3.2 Engineering Applications

This section uses various expected improvement algorithms to solve two engineering problems.
3.2.1 Injection Molded Disposable Camera

Injection molding is an integral component in the manufacturing of plastic parts. The quality of injection molded parts is highly dependent upon several conditions, such as melt and mold temperature, cooling time, packing time, and packing pressure. These various conditions is often set by combining prior knowledge with experimental trial and error. Unfortunately, this approach can be costly, time-consuming, and highly dependent on the molding operator’s skill and experience level. To set these conditions in a more efficient manner, process engineers have developed numerical models to simulate the injection molding process on a computer.

The injection molding application we will be focusing on was originally presented in Mulyana (2006) and Villarreal-Marroquin et al. (2010). We will use the simulation
software package MoldFlow Plastic Insight to determine the optimal manufacturing conditions for a Kodak disposable camera casing.

The inputs for this particular problem are

\[ x_1 = \text{mold temperature, } 20^\circ\text{C to } 70^\circ\text{C} \]
\[ x_2 = \text{packing time, } 1\text{ sec to } 10\text{ sec}. \]

There are other input variables that might be varied, but we will hold them fixed in this particular application. For example, we fix melt temperature at 200°C, packing pressure at 50 MPa, the injection time at 1 sec. The material used is Poly-Styrene manufactured by Dow Chemical USA under the trade name Styron 685D.

The goal is to preserve the design dimensions of the camera case during the injection molding process. In this particular study, we will focus attention on keeping the hole in the top right corner of the camera (see Figure 3.74) as square as possible, and as close to the original design dimensions as possible. This can be accomplished by minimizing the deviation (from the original design dimensions) in the length of a3 and minimizing the difference between the deviations in the length of a3 and b3. Therefore, the outputs that we wish to minimize are

\[ y_1(x) = |\Delta a_3|, \text{ measured in mm} \]
\[ y_2(x) = |\Delta a_3 - \Delta b_3|, \text{ measured in mm,} \]

where \( \Delta a_3 \) and \( \Delta b_3 \) are differences between the specified dimensions of a3 and b3 and the realized dimensions of a3 and b3 in the simulated injection molding process.

This computer experiment can take several minutes to evaluate \( y_1(\cdot) \) and \( y_2(\cdot) \), and there is no closed form expression for the output. This is precisely the sort of function for which the methods of Chapter 2 were devised. Because the output
requires somewhat long computations and some time-consuming manual calculations, we will not be able to test all combinations of improvement functions, dependence models, and $QI(\mathbf{x})$ methods. Specifically, we will only attempt to solve the problem using EMMI-Ind and ECI2-Ind, as these methods have proven to be effective on simpler examples. In both cases, our initial design will be a maximin Latin hypercube of size 9, and we will sequentially add 11 inputs points to the design. We will not be able to compare the estimated Pareto fronts and sets to the true Pareto front and set, since the truth is unknown and cannot be calculated over a dense grid in the input space.

Figure 3.75: Sequentially added inputs for the Kodak disposable camera problem using EMMI-Ind. The original inputs are magenta circles labeled 1-9, while the sequentially added inputs are black crosses numbered 10-20, with 10 being the first sequentially added point, and 20 being the last.

Figure 3.76: Sequentially added outputs for the Kodak disposable camera problem using EMMI-Ind. The original outputs are magenta circles labeled 1-9, while the sequentially added outputs are black crosses numbered 10-20, with 10 being the first sequentially added output, and 20 being the last.
The sequential designs for each method can be seen in Figures (3.75)-(3.78). The estimated Pareto set and front using the EMMI-Ind algorithm are

\[
P_{X-EMMI}^{20} = \begin{bmatrix}
51.0000 & 6.0000 \\
70.0000 & 8.0000 \\
43.7306 & 5.8854 \\
48.6313 & 5.9570 \\
37.9421 & 4.6665
\end{bmatrix}
\quad \text{and} \quad
P_{Y-EMMI}^{20} = \begin{bmatrix}
0.02 & 0.11 \\
0.05 & 0.05 \\
0.04 & 0.07 \\
0.03 & 0.10 \\
0.01 & 0.14
\end{bmatrix}.
\]

The estimated Pareto set and front using the ECI2-Ind algorithm are

\[
P_{X-ECI2}^{20} = \begin{bmatrix}
51.0000 & 6.0000 \\
70.0000 & 8.0000 \\
48.0700 & 5.5237 \\
44.9111 & 4.9435 \\
38.6569 & 5.4568 \\
28.5095 & 5.3406
\end{bmatrix}
\quad \text{and} \quad
P_{Y-ECI2}^{20} = \begin{bmatrix}
0.02 & 0.11 \\
0.05 & 0.05 \\
0.01 & 0.14 \\
0.00 & 0.18 \\
0.03 & 0.09 \\
0.04 & 0.06
\end{bmatrix}.
\]

Hypervolume measures for the estimated Pareto fronts, calculated using (.22, .22) as a reference point are \( I_H (P_{Y-EMMI}^{20}) = 0.0335 \) and \( I_H (P_{Y-ECI2}^{20}) = 0.0341 \).

From these results, it appears that the Pareto set includes almost the entire range of mold temperatures, with values as small as 28.5095°C and as large as 70.0000°C in the estimated Pareto sets. Pareto optimal packing times, however, seem to be concentrated between 5 sec and 8 sec. The upper and lower extremes for packing time appear to produce inputs that do a poor job of preserving the design dimensions.

Among the estimated Pareto sets that have been created, a design engineer would need to use their judgement regarding the relative importance of each output and cost of implementation to choose which design to use in the actual injection molding process.

Regarding the algorithms used (ECI2-Ind and EMMI-Ind), these results are somewhat surprising. In contrast to the example presented earlier, ECI2-Ind outperforms EMMI-Ind. Not only does the \( P_{Y-ECI2}^{20} \) have a larger hypervolume, but \( P_{Y-ECI2}^{20} \succeq P_{Y-EMMI}^{20} \). It appears that EMMI-Ind adds inputs whose outputs are
concentrated in the northwest region of the output space, while ECI2-Ind does a better job of generating wide range of outputs. This is unexpected, since EMMI-Ind is the more effective method in the example problems of Section 3.1.

There are several challenges built-in to this particular design problem. First, the initial design was very small, and only 20 total function evaluations were used in each expected improvement algorithm, mainly because of the computationally intensive nature of the function outputs. One could probably substantially improve upon the results if an initial 20 point maximin Latin hypercube design was used, as advocated in the computer experiments literature (see, for example, Loeppky et al. (2009)). Similarly, one could also improve results if a larger number of sequential inputs were
selected. Second, some of the Gaussian process model assumptions might be violated, as each output is bounded below by zero, and each output is rounded to only two decimal places, effectively forcing the continuous output to become discrete, ordinal output. The first modeling concern could be alleviated by modeling a transformation of the outputs (i.e., a natural logarithm) instead of the original units. It is not so clear how to fix the second complication, as essentially, we are observing a censored version of the simulator output for a given run (i.e., an output of 0.02 for $y_1(\cdot)$ could mean the true output is anywhere between 1.5 and 2.5). Perhaps adding some artificial measurement error (sometimes referred to as the “nugget”) could help with this issue.

Despite these issues, both EMMI-Ind and ECI2-Ind are reasonably effective at generating sequential designs that substantially improve upon the initial design. Therefore, it appears that these methods are reasonably robust to steep constraints on the computational budget and some violations to the Gaussian process modeling assumptions.

3.2.2 The Nowacki Beam

The multiobjective optimization problem considered here is a variant of the classic Nowacki beam engineering design problem (see Nowacki (1980)) described in Forrester et al. (2008) and implemented in MATLAB (available an online supplement for Forrester et al. (2008)). In contrast to the previous problems considered, this is a constrained multiobjective optimization problem.

The goal of the Nowacki beam problem is to design a mild steel tip-loaded encastre cantilever beam of minimum cross-sectional area and lowest bending stress subject
to a number of constraints. The inputs and their bounds are

\[ x_1 = h = \text{height of the beam, 20 mm to 250 mm} \]
\[ x_2 = b = \text{breadth of the beam, 10 mm to 50 mm}, \]

and the problem assumes that the beam length is fixed at \( l = 1.5 \text{ m} \) and subject to tip load of \( F = 5 \text{ kN} \). The outputs to be minimized are

\[ y_1(x) = x_1 x_2 = \text{cross-sectional area, measured in } \text{m}^2 \]
\[ y_2(x) = \text{beam bending stress, measured in } \text{Pa}. \]

The constraints functions are

\[ y_3(x) = \text{tip deflection, measured in } \text{mm} \]
\[ y_4(x) = \text{direct bending stress, measured in } \text{MPa} \]
\[ y_5(x) = \text{sheer stress, measured in } \text{MPa} \]
\[ y_6(x) = \text{breadth to height ratio} \]
\[ y_7(x) = \text{failure force for twist buckling, measured in } \text{Pa}. \]

The outputs and bounds of the constraint functions also depend on the material properties of mild steel, which are set to be: yield stress of \( \sigma_Y = 240 \text{ MPa} \), Young’s modulus \( E = 216.62 \text{ GPa} \), \( \nu = 0.27 \), and a shear modulus of \( G = 86.65 \text{ GPa} \). The constraints on \( y_3(\cdot), \ldots, y_7(\cdot) \) are

\[ y_3(x) < 5\text{mm} \]
\[ y_4(x) < 6Fl/(bh^2) \]
\[ y_5(x) < 3F/(2bh) \]
\[ y_6(x) < 10 \]
\[ y_7(x) > 2F. \]
Heat maps of $y_1(\cdot)$ and $y_2(\cdot)$ over the feasible region can be found in Figure 3.79 and Figure 3.80.

Figure 3.79: Feasible $y_1(\cdot)$ (cross-sectional area in mm$^2$).

Because the outputs for the Nowacki beam problem are not expensive to evaluate, we can $y_1(\cdot), \ldots, y_7(\cdot)$ over a dense grid of inputs, and use these values to construct an approximation to the true constrained Pareto set and front. Plots of the Pareto front and set are given in Figure 3.81 and Figure 3.82. The Pareto front is continuous
and convex, and the Pareto set is comprised of the “northwest” edge of the feasible region of the input space.

We will approximate the Pareto front and set using the expected improvement algorithm outlined in Section 2.1 with the modified improvement functions for constrained multiobjective optimization described in Section 2.1.2. For this particular problem, approximation of the Pareto front can be time consuming, since we must fit a model for seven functions (2 outputs and 5 constraints). Therefore, we will only run each method once, and we will restrict ourselves to the independence case only, since

Figure 3.80: Feasible $y_2(\cdot)$ (beam bending stress in MPa).
obtaining maximum likelihood estimates for a nonseparable dependence model requires us to estimate all \(7(6)/2 = 21\) off diagonal elements of the symmetric matrix \(A\). We will start each run with a 20 point maximin-LHD constructed via the MATLAB function `bestlh.m` (freely available in the online supplement for Forrester et al. (2008)), and sequentially add twenty more points via various via expected improvement algorithms with various choices for \(QI(x)\). All scaling dependent improvement criteria require some sort of empirical scaling of the output. For all scaling dependent method, with the exception of EHI1 and EHI2, if we have observed \(y(x_1), \ldots, y(x_n)\) then we scale all outputs as

\[
y^\text{scaled}_i(x_j) = \frac{y_i(x_j) - y^\text{cmin}_i}{y^\text{cmax}_i - y^\text{cmin}_i},
\]  

(3.2.1)

where \(y^\text{cmin}_i\) and \(y^\text{cmax}_i\) are the respective minimum and maximum of the observed outputs in the \(i^{th}\) dimension that have satisfied the constraints. For EHI1 and EHI2,
if we have observed $y(x_1), \ldots, y(x_n)$ then we scale all outputs as

$$y_i^{scaled}(x_j) = \frac{y_i(x_j) - y_i^{\min}}{y_i^{\max} - y_i^{\min}},$$

(3.2.2)

where $y_i^{\min}$ and $y_i^{\max}$ are the respective minimum and maximum of the of all observed outputs in the $i^{th}$ dimension, not only those that satisfied the constraints. The reference point for $I_H$ after scaling is $(1, 1)$, which corresponds to the unconstrained maximum among each dimension of the observed outputs.

In all cases, the scaling choice and reference point choice were chosen according to the method which produced the most favorable results. In the discussion, we will show how these choices can have a large effect on the estimated Pareto front.

**Results**

Because the Nowacki beam problem has a two-dimensional output space ($m = 2$), we can use both Pareto set quality indicators and visualization to assess the performance of the various improvement criteria. Letting $P_{40}^{\mathcal{Y}}$ denotes the approximated Pareto Front based on all 40 function observations, Table 3.12 lists the values of $I_{\epsilon^*}(P_{\mathcal{Y}}, P_{40}^{\mathcal{Y}})$ for Pareto front approximations created by several different $QI(\cdot)$ functions. Smaller values represent better approximations to the true Pareto front. The hypervolume indicator of the various Pareto front approximations was also computed using $R = (0.0125, 1.13 \times 10^8)$ (the constrained maximum of each input) as the reference point; larger values of the hypervolume indicator represent better approximations. These values can also be found in Table 3.12. The true Pareto front and Pareto set and various Pareto front and Pareto set approximations were plotted in Figures 3.84 - 3.88 to allow visual inspection of the approximations; the spread of an approximation and its closeness to the the true front and set are of particular interest.
Table 3.12: Summary of quality indicators for the twelve expected improvement algorithms for the Nowacki beam problem. Algorithms were run only once for each improvement criteria, and independent output models were used. The hypervolume indicator uses the constrained marginal maximums of the two outputs (0.0125, $1.13 \times 10^8$) as the reference point $R$.

<table>
<thead>
<tr>
<th>Method</th>
<th>$I_H(P_{y}^{40})$</th>
<th>$I_{e^*}(P_{y}, P_{y}^{40})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMMI</td>
<td>0.7730</td>
<td>0.0491</td>
</tr>
<tr>
<td>KEI</td>
<td>0.7622</td>
<td>0.0635</td>
</tr>
<tr>
<td>PI</td>
<td>0.6914</td>
<td>0.1322</td>
</tr>
<tr>
<td>EMAX1</td>
<td>0.7704</td>
<td>0.0645</td>
</tr>
<tr>
<td>ECI1</td>
<td>0.6716</td>
<td>0.1598</td>
</tr>
<tr>
<td>ECI2</td>
<td>0.7675</td>
<td>0.0647</td>
</tr>
<tr>
<td>EHI1</td>
<td>0.7585</td>
<td>0.0765</td>
</tr>
<tr>
<td>EHI2</td>
<td>0.7274</td>
<td>0.1136</td>
</tr>
<tr>
<td>EWHI1</td>
<td>0.7580</td>
<td>0.1046</td>
</tr>
<tr>
<td>EWHI2</td>
<td>0.7485</td>
<td>0.1205</td>
</tr>
<tr>
<td>EMMI2</td>
<td>0.7682</td>
<td>0.0649</td>
</tr>
<tr>
<td>KEI1</td>
<td>0.7653</td>
<td>0.0563</td>
</tr>
<tr>
<td>EECI1</td>
<td>0.7684</td>
<td>0.0629</td>
</tr>
<tr>
<td>EECI2</td>
<td>0.6560</td>
<td>0.1598</td>
</tr>
</tbody>
</table>
From the results in Table 3.12, it appears that EMMI performs best. This particular improvement criterion has both the highest hypervolume indicator (0.7730) and lowest binary-$\epsilon$ indicator (0.491). KEI, EMAX1, ECI2, EECI1, EMMI2, and KEI1 also appear very favorable with regard to the two performance metrics, as they all have hypervolumes greater than 0.76 and binary-$\epsilon$ indicators less than 0.07. While not performing poorly, the methods based on the hypervolume and weighted hypervolume indicator (EHI1, EHI2, EWHI1,EWHI2) all appear to be inferior to the six previously mentioned methods. PI, ECI1, and EECI2, three of the five scale invariant methods, perform the worst in terms of both metrics by a rather large margin.

Visual examination of the plots of the estimated Pareto fronts and sets in Figures 3.84-3.88 yields similar conclusions. EMMI appears to do the best job of adding points at or very near the true Pareto front and set, and it also appears to space these points rather evenly. EMAX1 and EMMI2, while still producing good estimates, seem to have less spread (particularly in the case of the Pareto set calculated using EMAX1) and more inputs that are a considerable distance from the true Pareto front and set. Among the scaling invariant methods, ECI2 and EECI1 produce the most visually appealing plots. All but the “eastern” most parts of the Pareto front and set are well-represented by the approximations, and the approximations have a rather uniform spread. PI and ECI1 perform poorly, as approximations produced by both methods show substantial clumping. It appears that these approaches are more concerned with adding inputs that are far away from the infeasible region of the input space, rather than adding points that are near the Pareto set. The results for EECI2 also appear disappointing, as the estimated front and set only contain six points, and all points are a considerable distance from the true Pareto objectives.
Somewhat surprisingly, KEI produces results that appear just as visually appealing, if not more visually appealing, than those produced by EMMI. Its method 2 counterpart, KEI, produces a well-spaced Pareto front, but an inspection of the estimated Pareto set shows that many of the sequentially added points are sub-optimal, as they are not in or very close to the true Pareto set. Much like the performance metrics, the plots of the hypervolume-based criteria fall somewhere in the middle of the pack. The plots of the estimated fronts and sets, while not as bad as PI and ECI1, show less uniformity, less spread, and more points that are a considerable distance from the true Pareto front than the other scaling dependent methods.

Discussion

A constrained problem adds some unique challenges for scaling dependent methods, and methods that require a reference point (EHI1 and EHI2). How should one scale outputs and calculate reference points based solely on the observed function evaluations if no (a priori) scaling information is available? Recall, we could scale based on the entire set of observed function evaluations (see (3.2.2)) or only based on those that meet the constraints (see 3.2.1). Unfortunately, such choices can have a large effect on the quality of the estimated Pareto front.

As an example, consider EHI1, KEI, and EMMI. We will use both scaling techniques described earlier in (3.2.1)-(3.2.2) in our expected improvement algorithms. For EHI1, the reference point for \( I_{H}(\cdot) \) after scaling is \((1, 1)\). When using (3.2.2), this corresponds to the unconstrained maximum among each dimension of the observed outputs. When using (3.2.1), this corresponds to the constrained maximum among each dimension of the observed outputs. We have plotted the estimated Pareto fronts
Figure 3.83: Comparison of Pareto front approximations using either scaling as defined in (3.2.2) or (3.2.1). Estimated Pareto fronts using EHI1 with (3.2.2) so the observed constrained maximums are the reference point (top left), EHI1 with (3.2.1) (top right) so the observed unconstrained maximums are the reference point, EMMI with (3.2.2) (middle left), EMMI with (3.2.1) (middle right), KEI with (3.2.2) (bottom left), and KEI with (3.2.1) (bottom right).
in Figure 3.83. Judging by these plots, there is a noticeable difference in the estimated Pareto front. For KEI, (3.2.2) appears to lead to fewer points on the Pareto front, most notably three points that are a considerable distance from the front. For EHI1, there seems to be a trade-off between the two approaches. Using (3.2.1), it appears that all points on the estimated Pareto front are at or near the true Pareto front. However, the marginal extremes in both dimensions are not represented at all by the estimated Pareto front. Using (3.2.2), the extremes are better represented for the bending stress output and there is more spread. However, some points on the estimated Pareto front (specifically the fifth from the top of the plot) are a considerable distance from the true front. For EMMI, (3.2.2) appears to produce a Pareto front approximation with slightly less spread, and slightly fewer points on true front as (3.2.1). But, these differences are not nearly as big as those seen using KEI and EHI1.

The point of this discussion is not to advocate a particular scaling approach or, in the hypervolume based methods, to recommend a particular reference point. Rather, the goal is just to show that methods that rely on the scaling of the output and external parameters (such as reference points) need to be used with caution, because the results can vary greatly based on these somewhat arbitrary choices.

The evidence of the sensitivity to the choice of scaling ((3.2.1) or (3.2.2)) makes an effective scaling invariant method, such as ECI2 or EECI1, look extremely attractive. Moreover, methods that require additional information, particular the hypervolume based methods (EHI1 and EHI2) look much less desirable for constrained problems that contain little prior information, as the choice of the reference point can have a large effect on the quality of the estimated Pareto front.
Figure 3.84: Estimated Pareto sets (left) and estimated Pareto fronts (right) using EMMI (top), EMAX1 (middle) and EMMI2 (bottom). The blue line in each plot represent the true Pareto set/front.
Figure 3.85: Estimated Pareto sets (left) and estimated Pareto fronts (right) using EHI1 (top), EWHI1 (middle) and KEI (bottom). The blue line in each plot represent the true Pareto set/front.
Figure 3.86: Estimated Pareto sets (left) and estimated Pareto fronts (right) using EHI2 (top), EWHI2 (middle) and KEI1 (bottom). The blue line in each plot represents the true Pareto set/front.
Figure 3.87: Estimated Pareto sets (left) and estimated Pareto fronts (right) using PI (top), ECI1 (middle) and ECI2 (bottom). The blue line in each plot represent the true Pareto set/front.
Figure 3.88: Estimated Pareto sets (left) and estimated Pareto fronts using EECI1 (top) and EECI2 (bottom). The blue line in each plot represent the true Pareto set/front.
CHAPTER 4

ESTIMATION OF SENSITIVITY INDICES BASED ON A GAUSSIAN PROCESS MODEL WITH A POLYNOMIAL MEAN

In this chapter, our focus will shift from multiobjective optimization to sensitivity analysis. Rather than having multiple output computer code, we will now focus on the situation where there is only one output, and the goal is to determine which inputs are most responsible for the variation in the computer code. This is done by computing sensitivity indices, which were introduced in Section 1.7. In Section 4.1, we introduce the Gaussian process (GP) model that we will use to emulate our computer code. Unlike previous research, we will assume that the Gaussian process has a mean that is a polynomial function of the inputs instead of a scalar. In addition to allowing us to model known large scale trends, such an assumption is also an integral part of some recent methods for emulating computer experiments with very large designs. In Section 4.2 and Section 4.3, we will explicitly detail how to compute estimates of sensitivity indices in the polynomial-mean Gaussian process framework for three different correlation functions. This research reflects the first time that such calculations have been made with the compactly supported Bohman correlation function, and it is also the first time that these calculations have been performed with
an arbitrary polynomial mean structure. In Section 4.4, we will present two examples that demonstrate the effectiveness of the proposed methods for estimating sensitivity indices.

### 4.1 Gaussian Processes with a Polynomial Mean

Here, we will present the Gaussian process model used throughout the remainder of the chapter. We will also discuss how the model can be fit in both the frequentist and Bayesian setting, and how it can be used to obtain predictions of the computer code. Since we are making the somewhat nonstandard polynomial mean assumption, we also present some motivating examples to show why such assumptions can be useful.

#### 4.1.1 Statistical Model

Denote the deterministic but unknown output of computer code by the function $y(x) = y(x_1, \ldots, x_d)$. Assume the $d$-dimensional cube, $\mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_d = [l_1, u_1] \times \ldots \times [l_d, u_d]$, is the input space. We model $y(x)$ as a draw from a Gaussian process $Y(x)$ with a polynomial mean

$$m_Y(x) = \mathbb{E}_p[Y(x)] = \sum_{k_1=0}^{n_{k_1}} \cdots \sum_{k_d=0}^{n_{k_d}} \beta_{k_1 \ldots k_d} \prod_{i=1}^d x_i^{k_i},$$

(variance $1/\lambda_Y$, and covariance function

$$\text{Cov}_p[Y(x_1), Y(x_2)] = \frac{1}{\lambda_Y} R_Y(x_1, x_2; \theta) = \frac{1}{\lambda_Y} \prod_{k=1}^d R_Y(x_{1,k}, x_{2,k}; \theta_k)$$

where $\theta = (\theta_1, \ldots, \theta_d)$ is a vector of parameters of the correlation function. Here $E_p$ and $\text{Cov}_p$ denote expectation and covariance with respect to the GP $Y(\cdot)$, $x_1$ and $x_2$ denote two input sites, and $x_{1,k}$ and $x_{2,k}$ denote the values of the $k^{th}$ input of these
two inputs, respectively. In particular, assume the code calculation at a specific input site \( x \) can be viewed as a realization of

\[
Z_{\text{sim}}(x) = Y(x) + \epsilon_{\text{sim}}(x),
\]

where \( \epsilon_{\text{sim}}(x) \) is a white noise process with mean zero and variance \( 1/\lambda_\epsilon \) and is independent of \( Y(x) \). The term \( \epsilon_{\text{sim}}(x) \) should be thought of as explicitly modeling non-deterministic behaviour of the computer output or enhancing numerical stability in the estimation of the correlation parameters. Then, the vector of observed outputs \( Z_{\text{sim}} = (Z_{\text{sim}}(x_1), \ldots, Z_{\text{sim}}(x_n))^\top \) has a mean vector \( m_{\text{sim}}^Z = [m^Y(x_1), \ldots, m^Y(x_n)]^\top \) and covariance matrix

\[
\Sigma_{\text{sim}}^Z = \frac{1}{\lambda_Y} R + \frac{1}{\lambda_\epsilon} I_n
\]

where the \((i, j)^{th}\) element of the \( n \times n \) matrix \( R \) is \( R^Y(x_i, x_j; \theta) \) and \( I_n \) is the \( n \times n \) identity matrix.

4.1.2 Prediction

If we have observed \( Z_{\text{sim}} = (Z_{\text{sim}}(x_1), \ldots, Z_{\text{sim}}(x_n))^\top \), we can then use the Gaussian process model assumption as a basis for prediction of \( y(x) \) from both a frequentist and Bayesian point of view. To simplify the notation, we will express the mean as

\[
m^Y(x) = f^\top(x) \beta,
\]

where

\[
\beta = [\beta_{0,...,0}, \beta_{0,...,1}, \ldots, \beta_{n_{d-1},n_d}, \beta_{n_{d-1},n_d}]^\top,
\]

\[
f(x) = \left[1, x_d, \ldots, \left(\prod_{i=1}^{d-1} x_i^{n_i}\right) x_d^{n_d-1}, \prod_{i=1}^{d} x_i^{n_i}\right]^\top.
\]

Also, let \( F = [f(x_1), \ldots, f(x_n)]^\top \) and \( c = \frac{1}{\lambda_Y} r \), where the \( i^{th} \) element of the \( n \times 1 \) vector \( r \) is \( R^Y(x, x_i; \theta) \), and let \( \sigma^2 \equiv \text{Var}[Y(x)] = 1/\lambda_Y + 1/\lambda_\epsilon \). The likelihood
function in the Gaussian process model is

\[
L(\beta, \lambda_Y, \lambda_\epsilon, \theta|Z_{sim}) = \frac{1}{(2\pi)^{n/2}|\Sigma_{sim}^{Z}|^{1/2}} \exp \left\{ -\frac{1}{2} (Z_{sim} - F\beta)^\top (\Sigma_{sim}^{Z})^{-1} (Z_{sim} - F\beta) \right\},
\]

(4.1.3)

In a frequentist setting, the best linear unbiased predictor (BLUP) of \(Y(x)\) when \(\theta, \frac{1}{\lambda_Y}, \text{ and } \frac{1}{\lambda_\epsilon}\) are known can be shown to be

\[
\hat{y}(x) = f^\top(x)\hat{\beta} + c^\top (\Sigma_{sim}^{Z})^{-1} (Z_{sim} - F\hat{\beta})
\]

where

\[
\hat{\beta} = \left( F^\top (\Sigma_{sim}^{Z})^{-1} F \right)^{-1} F^\top (\Sigma_{sim}^{Z})^{-1} Z_{sim},
\]

and \(\hat{y}(x)\) has mean square prediction error (MSPE)

\[
s^2(x) = \frac{1}{\lambda_Y} - \left[ f^\top(x), \ c^\top \right] \left[ \begin{array}{cc} 0 & F^\top \\ F & \Sigma_{sim}^{Z} \end{array} \right]^{-1} \left[ \begin{array}{c} f(x) \\ c \end{array} \right].
\]

When \(\theta, \frac{1}{\lambda_Y}, \text{ and } \frac{1}{\lambda_\epsilon}\) are not known, we plug maximum likelihood estimates of these parameters into the BLUP and MSPE. Here, we will summarize the approach taken in Huang et al. (2006) for calculating MLEs of the unknown parameters. First let

\[
R_{sim}^{Z} = \Sigma_{sim}^{Z} / \left( \frac{1}{\lambda_Y} + \frac{1}{\lambda_\epsilon} \right)
\]

be the correlation matrix of \(Z_{sim}\). The \((i, j)\)th off-diagonal elements of \(R_{sim}^{Z}\) are equal to \(R^Y(x_1, x_2; \theta)^{1/\lambda_Y + 1/\lambda_\epsilon}\). Then, the MLE of \(\theta\) and the ratio \(g \equiv \frac{1/\lambda_Y}{1/\lambda_Y + 1/\lambda_\epsilon}\) can be found by maximizing the logarithm of the profile likelihood function

\[
\ell(\theta, g|Z_{sim}) = -\frac{1}{n} \log (|R_{sim}^{Z}|) - \log (\hat{\sigma}^2)
\]

where

\[
\hat{\sigma}^2 = \frac{1}{n} (Z_{sim} - F\hat{\beta})^\top (R_{sim}^{Z})^{-1} (Z_{sim} - F\hat{\beta}).
\]

228
Plugging the MLE of $\theta$ and $g$ into $\hat{\sigma}^2$ yields the MLE for $\sigma^2$. We can then solve for the MLEs of $\frac{1}{\lambda_Y}$ and $\frac{1}{\lambda_\epsilon}$ using the known MLEs of $\sigma^2$, $\theta$ and $g$. In the case where there is no measurement error ($1/\lambda_\epsilon = 0$), $g = 1$ and no longer needs to be estimated, and the MLE of $\frac{1}{\lambda_Y}$ is simply $\hat{\sigma}^2$.

In the fully Bayesian approach, we assume a prior distribution $[\beta, \lambda_Y, \lambda_\epsilon, \theta]$ on the unknown parameters, and use the posterior distribution

$$
[Y(x)|Z_{sim}] = \int [Y(x), \beta, \lambda_Y, \lambda_\epsilon, \theta|Z_{sim}] d\beta d\lambda_Y d\lambda_\epsilon d\theta
$$

$$
= \int [Y(x)|\beta, \lambda_Y, \lambda_\epsilon, \theta, Z_{sim}] [\beta, \lambda_Y, \lambda_\epsilon, \theta|Z_{sim}] d\beta d\lambda_Y d\lambda_\epsilon d\theta
$$

to get both predictions and a measure of uncertainty for $Y(x)$. While the resulting posterior distribution for $Y(x)$ takes into account parameter uncertainty, we cannot derive it analytically and it therefore must be sampled from via MCMC. For more details regarding the Bayesian approach to prediction of computer experiments, see Chapter 4.1 of Santner et al. (2003) and Higdon et al. (2008).

### 4.1.3 Motivating the Polynomial Mean Assumption

Much of the computer experiments literature assumes that Gaussian process $Y(\cdot)$ modeling $y(\cdot)$ has a constant mean $m_Y(x) = \beta$ (see, for example, Sacks et al. (1989), Welch et al. (1992), Jones et al. (1998), and Higdon et al. (2008)). The reasons for this assumption include a lack of prior knowledge of trends in the data, simpler elicitation of priors distributions in a Bayesian setting, more efficient maximum likelihood estimation in a frequentist setting, and the fact that in many cases, a constant mean process has enough predictive power that adding more complicated regression terms is unnecessary. However, in some situations, it can be advantageous to use a more complex mean structure. Here, we detail two such cases.
Existence of Trends in $y(\cdot)$

Suppose that we have reason to believe there is a strong underlying trend (linear, quadratic, etc.) in the function being modeled. If we incorporate this trend into our GP model, we can greatly improve the predictive power, particularly when extrapolating outside the range of a small initial design. (By small, we mean a design that has much less than ten function evaluations per input dimension, a generally accepted informal rule for effective initial designs advocated by Loeppky et al. (2009).) Here, we will illustrate the advantages of using a non-constant mean in this situation via two simple examples.

First, consider the following function over $[0, 1]$:

$$y(x) = \sin(8\pi x) + 16\pi(x - 0.5)^2,$$  \hspace{1cm} (4.1.4)

which while not a polynomial, has a strong polynomial trend. Suppose that we wish to model this function, and our initial 7-point design is

$$(0.1000, 0.2429, 0.3857, 0.5286, 0.6714, 0.8143).$$

We will fit two GP models: one with a constant mean $\beta$, and another with a quadratic mean $\beta_0 + \beta_1 x + \beta_2 x^2$. We will then get 100 predictions $\hat{y}(x)$ at

$$(x_1 = 0, x_2 = \frac{1}{99}, \ldots, x_{99} = \frac{98}{99}, x_{100} = 1).$$

Judging by plots of the predicted function values for each mean structure in Figure 4.1, the quadratic mean model appears to yield better predictions, particularly for values larger than 0.8143 and smaller than 0.1. Additionally, if we calculate the empirical root mean square prediction error (ERMSPE), defined as

$$ERMSPE(\hat{y}(\cdot)) = \sqrt{\frac{\sum_{i=1}^{100}(\hat{y}(x_i) - y(x_i))^2}{100}},$$

230
for both models, we get 1.5361 for the constant mean model and 0.9160 for the quadratic mean model. Since lower ERMSPE indicates better predictions, this metric also supports the superiority of the quadratic mean model in this particular example.

Next, consider the following function over $[0, 1]^2$:

$$y(x) = (x_1 - 0.5)^2 + (x_2 - 0.5)^2 + (x_1 - 0.5)(x_2 - 0.5) + 0.05 \sin(4\pi x_1) + 0.05 \sin(4\pi x_2).$$  

(4.1.5)

Notice that the function contains a strong quadratic component. We start by first evaluating (4.1.5) at an initial 10-point maximin Latin hypercube design, which is plotted in Figure 4.2. Next, we get predictions at the 2500 points comprising a dense $50 \times 50$ grid in $[0, 1]^2$. Just as in the previous example, we calculate the ERMSPE (the only difference is that we are now averaging over 2500 points instead of 100) for
a model with a constant mean and a quadratic mean

\[ \beta_0 + \beta_{10}x_1 + \beta_{01}x_2 + \beta_{20}x_1^2 + \beta_{02}x_2^2 + \beta_{11}x_1x_2. \]  

(4.1.6)

We will also plot both predicted surfaces, which we will denote as \( \hat{y}_c(\cdot) \) for the constant mean model, and \( \hat{y}_q(\cdot) \) for the quadratic mean model. Additionally, we will plot \((\hat{y}_c(\cdot) - y(\cdot))^2 - (\hat{y}_q(\cdot) - y(\cdot))^2\), the difference between the squared prediction using the constant mean GP and the squared prediction error using the quadratic mean GP.

Just as we saw in the previous example, the quadratic mean model appears to have an edge in terms of predictive power. The constant mean model has an ERMSPE of 0.0839 compared to an ERMSPE of 0.0515 in the quadratic mean model. Judging by the plots in Figure 4.3, this appears to be mainly because of the quadratic models ability to extrapolate in the regions near (0,0) and (1,1) where there is no training.
Figure 4.3: *Top Left:* Plot of $y(x_1, x_2)$ in (4.1.5) over $[0, 1]^2$. *Bottom Left:* Predictions using a constant mean model based on 10-point design. *Bottom Right:* Prediction using a GP with mean 4.1.6 based on 10-point design. *Top Right:* Squared prediction errors of the quadratic mean model subtracted from the squared prediction errors in the constant mean model computer over a $50 \times 50$ grid in $[0, 1]^2$. 

233
data. The quadratic mean GP continues to push predictions upward as we move closer to these boundaries, while the constant mean GP is pulling these predictions down to the estimated $\beta$.

In this subsection, we have shown that if trends in a function are known a priori, we can greatly increase our prediction accuracy if we explicitly model these trends. These gains are most substantial in regions where we are extrapolating outside of the range of the original design. Notice, however, that we have restricted ourselves to Gaussian processes with means that are polynomial functions of the inputs. Is such an assumption too restrictive? While there certainly could exist trends in $y(\cdot)$ that are not polynomial functions of the input variables, polynomials are generally flexible enough to model a wide variety of surfaces, and non-polynomial trends could be approximated with polynomials using a Taylor series approximation. Moreover, restricting ourselves to polynomials makes the integral calculations necessary for computing sensitivity indices in Section 4.2 analytically tractable.

**Modeling Computer Experiments with Large Designs**

A weakness of the Gaussian process model presented in Section 4.1.1 is that evaluation of the likelihood function 4.1.3 requires inversion of the $n \times n$ matrix $\Sigma_{Z_{\text{sim}}}$. If the design $(x_1, \ldots, x_n)$ of the computer experiment is large, this matrix inversion can become prohibitively time consuming, and it can also become numerically unstable, as $\Sigma_{Z_{\text{sim}}}$ can become near-singular if the design is sufficiently dense. Thus, fitting the GP model can become computationally infeasible in both the Bayesian and frequentist setting for large $n$, since both require a large number of likelihood evaluations to either calculate MLEs or to sample from the posterior predictive process via MCMC.
To overcome these difficulties, Kaufman et al. (2010) has introduced some nonstandard modeling assumptions, which include a strong polynomial mean structure, for easing the computational burden when $n$ is large. In the following, we will summarize their approach.

First, the correlation functions $R^Y(x_{1,k}, x_{2,k}; \theta_k)$, $\theta_k > 0$, used in the product covariance (4.1.2) are chosen so that they have compact support. This means that $R^Y(x_{1,k}, x_{2,k}; \theta_k) = 0$ if $|x_{1,k} - x_{2,k}| > \theta_k > 0$. Some examples of compactly supported correlation functions are the Bohman and cubic correlation functions, which are defined in Section 4.3. Notice that the commonly used Gaussian correlation function ($\exp \{-\theta_k(x_{1,k} - x_{2,k})^2\}$ for $\theta > 0$) does not have compact support. Using correlation functions with compact support can introduce zeros into the covariance matrix $\Sigma_{sim}^Z$, which means likelihood calculations can exploit computationally efficient sparse matrix algorithms. Barry and Pace (1997) and Kaufman et al. (2010) both show that such methods can dramatically reduce the time needed to evaluate the likelihood in large datasets. For details of how some sparse matrix algorithms are implemented in MATLAB, we refer the reader to Gilbert et al. (1991).

On their own, compactly supported correlation functions are not enough to resolve the computational issues introduced by large designs. One, if the correlation lengths $\theta_1, \ldots, \theta_n$ are too large, there could be little or no sparsity in $\Sigma_{sim}^Z$. Two, if the level of sparsity is sufficient, the GP model might have little predictive power since only points that are relatively close to $x$ will be used to construct a predictor for $y(x)$. Kaufman et al. (2010) partially resolves these issues by attempting to model the mean of the GP with polynomial functions of the input, rather than just assuming it is a constant. For example, one particular simulation example with two inputs employs
a mean function that is a fifth degree polynomial in each dimension, as well as all two variable interactions in which the sum of the powers is less than or equal to five (i.e., the regression terms are $1, x_1, \ldots, x_5, x_2, \ldots, x_2^5, x_1^2 x_2, x_1 x_2, \ldots, x_4^4 x_2$). This has the effect of decreasing the magnitude of $\theta_1, \ldots, \theta_n$ and increasing the sparsity of $\Sigma_{sim}^Z$ without simultaneously destroying the predictive power of the Gaussian process model. One must be careful not to include too many regression terms, however, as calculation of the Gaussian likelihood requires inversion of $F^\top (\Sigma_{sim}^Z)^{-1} F$, which is not possible if $F$ does not have full column rank.

The final innovation of Kaufman et al. (2010) involves explicitly forcing a particular degree of sparsity on $\Sigma_{sim}^Z$ by limiting the correlation lengths. Since the authors take a fully Bayesian approach, this is accomplished via the prior distribution of $\theta$, which is assumed to be uniform over the space

$$T_C = \left\{ \theta \in \mathbb{R}^d : \theta_k \geq 0 \; \forall \; k \in \{1, \ldots, d\}, \sum_{k=1}^d \theta_k \leq C \right\}, \quad C > 0. \quad (4.1.7)$$

$C$ is chosen so that only a certain proportion $\alpha$ of the $\frac{n(n-1)}{2}$ off diagonal elements of $\Sigma_{sim}^Z$ can be nonzero. To do so, calculate

$$M_{i,j} \equiv \sum_{k=1}^d |x_{i,k} - x_{j,k}|$$

for each of the $\frac{n(n-1)}{2}$ pairs $(x_i, x_j)$ where $i > j$. Then, choose $C$ to be the $\lfloor \alpha \frac{n(n-1)}{2} \rfloor^{th}$ smallest value among the $M_{i,j}$’s. This ensures that, at most, $\alpha 100\%$ of the off-diagonal elements of $\Sigma_{sim}^Z$ are nonzero. Defining $T_C$ to be a $d-$dimensional triangle allows for some $\theta_k$’s to be large, which may be necessary if certain input dimensions show a high amount of correlation, while still controlling the degree of sparsity of the covariance matrix.
In a frequentist setting, as opposed to putting a uniform prior on \( \theta \) over \( T_C \), we restrict the parameter space of \( \theta \) to be \( T_C \) instead of \([0, \infty)^d\). Maximum likelihood estimates of \( \theta \) and \( g \) are then found by solving the constrained optimization problem

\[
\text{maximize} \quad \ell(\theta, g | Z_{\text{sim}}) = -\frac{1}{n} \log \left( |R_{\text{sim}}^Z| \right) - \log \left( \hat{\sigma}^2 \right) \quad (4.1.8)
\]

subject to \( \theta_k \geq 0 \ \forall \ k \in \{1, \ldots, d\}, \sum_{k=1}^d \theta_k \leq C, \ g > 0. \quad (4.1.9) \]

As an example in a frequentist setting, consider the borehole function, which is first mentioned in the computer experiments literature in Morris et al. (1993). This function models the flow of water through a borehole that is drilled from the ground surface through two aquifers. The eight inputs and their ranges are as follows:

\[
\begin{align*}
x_1 &= r = \text{radius of influence}, \ 100 \text{ to } 50,000 \text{ m} \\
x_2 &= r_w = \text{radius of borehole}, \ .05 \text{ to } .15 \text{ m} \\
x_3 &= T_u = \text{transmissivity of upper aquifer}, \ 63,070 \text{ to } 115,600 \text{ m}^2/\text{yr} \\
x_4 &= H_u = \text{potentiometric head of upper aquifer}, \ 990 \text{ to } 1,110 \text{ m} \\
x_5 &= T_l = \text{transmissivity of lower aquifer}, \ 63.1 \text{ to } 116 \text{ m}^2/\text{yr} \\
x_6 &= H_l = \text{potentiometric head of lower aquifer}, \ 700 \text{ to } 820 \text{ m} \\
x_7 &= L = \text{length of borehole}, \ 1,120 \text{ to } 1,680 \text{ m} \\
x_8 &= K_w = \text{hydraulic conductivity of borehole}, \ 9,855 \text{ to } 12,045 \text{ m/yr}. \\
\end{align*}
\]

The response variable \( y(x) \), the flow rate through the borehole in \( \text{m}^3/\text{yr} \), is determined by the equation

\[
y(x) = \frac{2\pi T_u (H_u - H_l)}{\ln(r/r_w) \left( 1 + \frac{2LT_u}{\ln(r/r_w) r_w^2 K_w + T_u / T_l} \right)}.
\]

Note that \( y(\cdot) \) ranges from a minimum around 10 \( \text{m}^3/\text{yr} \) to around 220 \( \text{m}^3/\text{yr} \).
Suppose that we try to fit a Gaussian process model to the borehole function with an initial 400 point maximin LHD with the standard constant mean assumption and Gaussian correlation function. There is no measurement error in this function so, ideally, we would like to fit a model with \(1/\lambda = 0\) so that the predictive surface interpolates at any \(x\) in the initial design. Unfortunately, as mentioned before, this requires inversion of the near-singular \(400 \times 400\) matrix \(\Sigma_{sim}\) with all nonzero elements. Obtaining MLEs is time consuming, the results are unstable, and the BLUP will be very poor if we try to fit an interpolating process to this data.

To partially alleviate these computational difficulties, JMP software adds a small amount of measurement error to the model, which alleviates the near-singularity of the covariance matrix. While such a measure increases the stability of the estimates and produces good predictions of \(y(x)\) (the ERMSPE at 200 validation points is only 1.1084), there are two drawbacks. One, since we are assuming there is measurement error that is not actually present, the resulting predictor no longer interpolates the original data. Two, this implementation still requires inversion of a dense (the off-diagonal elements are all nonzero) \(400 \times 400\) matrix, a computationally demanding task. In JMP, fitting this model and obtaining estimates of sensitivity indices takes over an hour on an Intel Core 2 Duo CPU 3.00 with 4 GB RAM.

If we model the borehole function with a polynomial mean that includes a fifth degree polynomial of each input with all interactions between the inputs (i.e., the regression terms are \(x_1, x_1^2, \ldots, x_1^5, \ldots, x_8, x_8^2, \ldots, x_8^5, x_1x_2, x_1x_3, \ldots, x_7x_8\)), the Bohman correlation function, and a restricted parameter space that only permits, at most, ten percent of the off-diagonal elements of \(\Sigma_{sim}\) to be nonzero, we can eliminate the drawbacks of JMP’s implementation with little sacrifice to predictive power. Specifically,
the forced sparsity makes inversion of $\Sigma_{sim}^Z$ much more efficient, and also eliminates the need to add artificial measurement error. Therefore, the resulting predictor is an interpolator. Our MPErK implementation (see Han et al. (2011)) took less than twenty minutes to fit the model and estimate sensitivity indices on the same machine, and the ERMSPE of the resulting predictor at the same 200 validation points was only 1.4757. While this is slightly bigger than the ERMSPE in JMP’s model, the difference is of little practical significance when one considers the range of the $y(\cdot)$ values. More details of this example will be presented in Section 4.4.2.

For some examples in a Bayesian setting, we refer the reader to Kaufman et al. (2010). Here, the authors use a simple 1-d example to show that the predictions and confidence bands using the preceding sparse implementation of the Bohman correlation function with a 5th degree polynomial mean are nearly indistinguishable from the predictions and confidence bands calculated using a power exponential correlation function ($\exp\{-\theta_k|x_{1,k} - x_{2,k}|^{\alpha_k}\}$ for $1 \leq \alpha_k \leq 2$ and $\theta_k > 0$) and a constant mean. They also successfully model a photometric red-shift simulator with four inputs and a 20,000 input design using the proceeding modeling strategies (which would be impossible using some of the standard methods), and they also discuss some data analytic strategies for choosing the polynomial mean function.

4.2 Estimation of Sensitivity Indices in the Gaussian Process Model

There are three methods proposed in the literature for estimating sensitivity indices in computer experiments. Recall from Section 1.7 that the calculation of $S_i$ and $T_i$ requires calculation of integrals of $y(\cdot)$. The first approach, employed in Saltelli (2002), Morris et al. (2006), Morris et al. (2008), and Saltelli et al. (2010),
is a sampling-based strategy in which the computer experiment is designed so that
the sample variance of the computer code output for particular groups of input are
unbiased estimators of the joint effect variances $V_{jS}^2$. Only output from the actual
computer experiment is used to construct the estimators. In the second approach, we
replace $y(\cdot)$ with the BLUP of $y(\cdot)$ in the various integrals, i.e.,

$$\hat{V}_{jS}^2 = Var_g[E_g[y(X)|X_S]].$$

(4.2.1)

This is the approach used by Sacks et al. (1989), Welch et al. (1992) and JMP
software. The third method is Bayesian and replaces $y(\cdot)$ by the process $Y(\cdot)$ in the
relevant integrals and uses the fact that, at least for Gaussian process models, the joint
distribution of the integrated process and the process evaluated at the training data
site is multivariate normal to calculate the posterior expected value of the integrated
process given the training data. In this case,

$$\hat{V}_{jS}^2 = E_p \{Var_g[E_g[Y(X)|X_S]] | Z_{sim}\},$$

(4.2.2)

where $E_p$ represents expectation with respect to the random Gaussian process. This is
the approach taken in Oakley and O’Hagan (2004), Gramacy and Taddy (2010), Moon
(2010), and implemented in the GPM/SA code developed by Los Alamos National
Laboratory. This is also the approach we will take in the section. Notice that the
last two estimates, unlike the first, both use the Gaussian process model assumption.
Moreover, these estimates depend upon the correlation function $R^Y(x_1, x_2; \theta)$ and
the mean function $m^Y(x)$ of $Y(\cdot)$. The advantage is that they can be calculated for
an arbitrary design.

Our estimates of $V_{jS}^2$, calculated via (4.2.2), will assume $Y(x)$ follows the distri-
butional assumptions from Section 4.1.1, i.e., $Y(x)$ is a GP with a mean that is a
polynomial function of the inputs and product correlation form. Specifically, we will assume that $R^Y(x_1, x_2; \theta)$ is a product of $d$ Gaussian, cubic, or Bohman correlation functions (see Section 4.3.1-4.3.3 for the definition of these functions). Additionally, we will assume that all mean, variance, and correlation parameters are known. In practice, these parameters are unknown, so we must either plug in estimates for them (calculated via maximum likelihood, REML, cross-validation, or some other estimation technique) or elicit prior distributions on them and integrate $\hat{V}_S^Y$ with respect to the posterior distribution $[\beta, \lambda_Y, \lambda_z, \theta | Z_{\text{sim}}]$. The implementation and examples in Section 4.4 plug in maximum likelihood estimates of these parameters. The MLEs can be calculated with either an unconstrained parameter space (i.e., $\theta \in (0, \infty)^d$) for all three correlation functions or with a constrained parameter space (i.e., $\theta \in T_C$, where $T_C$ defined in (4.1.7)) for the compactly supported Bohman and cubic correlation functions. Therefore, our method of estimation is best described as an empirical Bayesian approach that allows for one to use a constrained parameters space and computationally efficient sparse matrix methods if necessary (i.e., if the initial design is large).

We will also make the simplifying assumption that $\mathcal{X} = [0, 1]^d$ and each $g_k(\cdot)$ is uniform on $[0, 1]$. Also, similar to $E_p$, let $Cov_p$ represent covariance with respect to the random Gaussian process. Additionally, for any $v \times w$ dimensional matrix $M$, we will let $K = \int M \, dx$ denote the $v \times w$ matrix with $(i, j)$th element equal to the integral of the $(i, j)$th element of $M$ with respect to $x$, i.e., $[K]_{i,j} = \int [M]_{i,j} \, dx$.

Before providing detailed calculations, here’s how does our approach to computing (4.2.2) compares to the current literature:
1. Moon (2010) calculates (4.2.2) under the assumption that $Y(\cdot)$ has a constant mean $\beta$ and a Gaussian or cubic correlation function (but not a Bohman correlation function). Moon (2010) also assume that each $g_k(\cdot)$ is uniform. To deal with parameter uncertainty, Moon (2010) also uses maximum likelihood estimates, but the MLEs can only be calculated assuming $\theta \in (0, \infty)^d$. Therefore, our approach offers more flexibility in terms of the modeling assumptions for $Y(\cdot)$ and the estimation of correlation parameters.

2. GPM/SA software calculates (4.2.2) under the assumption that $Y(\cdot)$ has a constant mean $\beta$ and a Gaussian correlation function. The software also assume that each $g_k(\cdot)$ is uniform. GPM/SA takes a fully Bayesian approach, putting proper priors on all unknown parameters, and calculates the mean of (4.2.2) with respect to the posterior distribution of the parameters. Therefore, our approach offers more flexibility in terms of the modeling assumptions for $Y(\cdot)$, and can also calculate $\hat{V}^j_S$ for large designs via sparse matrix methods. The advantage of GPM/SA is that it provides estimates for $\hat{V}^j_S$ that account for parameter uncertainty.

3. Rather than assuming $Y(\cdot)$ is a GP with known parameters, Oakley and O’Hagan (2004) assumes only the correlation parameters $\theta$ are known and puts an improper prior on both the mean parameters and the process variance. Doing so, the posterior distribution of $Y(x)$ is a noncentral t-distribution. They derive integral expressions for the posterior means of $V_S$ for arbitrary $g(x)$, process means, and correlation functions, but do not detail any closed form expressions for these integrals. The examples in Oakley and O’Hagan (2004) are limited
to Gaussian correlation functions with either a constant mean or a mean that is a linear function of the inputs. Estimates for the correlation parameters are plugged in since no priors are elicited. Therefore, the approach of Oakley and O’Hagan (2004) has slightly different modeling assumptions, as priors are elicited on some of the unknown parameters. Also, unlike this thesis, the authors do not give explicit, analytic expressions for (4.2.2); instead, they provide general formulas, and leave the reader to work out the details for a specific correlation function and mean structure.

Now that the advantages and limitations of our approach are known, as well as how our research compares to previous research, we will present the details of the calculations necessary to compute $\hat{V}_S^j$.

Analogous to the joint effect function $j_S(x_S)$ defined in (1.7.8) for $y(\cdot)$,

$$J_S(x_S) = E_g[Y(X)|X_S = x_S] = \int_{X-S} Y(x_S, x_S) dx_S$$

(4.2.3)

can be viewed as the prior process for $j_S(x_S)$.

Viewed as a process with index $x_S$, $J_S(x_S)$ is a GP with mean function

$$m_S^j(x_S) \equiv E_p[J_S(x_S)]$$

$$= \int_{x_S} E_p[Y(x)] dx_S$$

$$= \int_{x_S} \sum_{k_1=0}^{n_{k_1}} \cdots \sum_{k_d=0}^{n_{k_d}} \beta_{k_1 \ldots k_d} \prod_{i=1}^{d} x_i^{k_i} dx_S$$

$$= \sum_{k_1=0}^{n_{k_1}} \cdots \sum_{k_d=0}^{n_{k_d}} \beta_{k_1 \ldots k_d} \prod_{j \notin S} (k_j + 1)^{-1} \prod_{i \in S} x_i^{k_i},$$

(4.2.4)

covariance function
\[ C_S^J(x_S, x'_S) \equiv \text{Cov}_p[J_S(x_S), J_S(x'_S)] \]
\[ = \text{Cov}_p \left[ \int Y(x)dx_{-S}, \int Y(x')dx'_{-S} \right] \]
\[ = \frac{1}{\lambda_Y} \int \int R^Y(x, x'; \theta)dx_{-S}dx'_{-S} \]
\[ = \frac{1}{\lambda_Y} \prod_{k \notin S} \left[ \int_0^1 \int_0^1 R^Y(x_k, x'_k; \theta_k)dx_kdx'_k \right] \prod_{k \in S} R^Y(x_k, x'_k; \theta_k) \]
\[ = \frac{1}{\lambda_Y} \prod_{k \notin S} \text{dbint}(0, 1; \theta_k) \prod_{k \in S} R^Y(x_k, x'_k; \theta_k), \tag{4.2.5} \]
and variance function
\[ C_S^J(x_S, x_S) \equiv \text{Cov}_p[J_S(x_S), J_S(x_S)] = \frac{1}{\lambda_Y} \prod_{k \notin S} \text{dbint}(0, 1; \theta_k) \tag{4.2.6} \]
since \( R^Y(x_k, x_k; \theta_k) = 1 \). The integral \text{dbint} was originally derived for the Gaussian and cubic correlation functions in Moon (2010). In Section 4.3, we present these derivations, and also provide new derivations of \text{dbint} for the Bohman correlation function.

Now \([J_S(x_S), Z_{\text{sim}}]^\top\) has the multivariate normal distribution
\[ N \left[ \begin{pmatrix} m^J_S(x_S) \\ m^Z_{\text{sim}} \end{pmatrix}, \begin{pmatrix} C_S^J(x_S, x_S) & \text{Cov}_p[J_S(x_S), Z_{\text{sim}}] \\ \text{Cov}_p[J_S(x_S), Z_{\text{sim}}]^\top & \Sigma_Z \end{pmatrix} \right]. \tag{4.2.7} \]
Define \( J^p_S(x_S) \) to be the process whose distribution is that of the posterior predictive process of \( J_S(x_S) \) given \( n \) code calculations \( z_{\text{sim}} = (z_{\text{sim}}(x_1), \ldots, z_{\text{sim}}(x_n))^\top \) and whose marginal distribution can be derived from (4.2.7). Further \( J^p_S(x_S) \) is a GP (see Johnson and Wichern (1998), pages 160–161) with mean function
\[ m^p_S(x_S) = m^J_S(x_S) + \text{Cov}_p[J_S(x_S), Z_{\text{sim}}] (\Sigma_Z^{-1}_{\text{sim}}) (z_{\text{sim}} - m^Z_{\text{sim}}) \tag{4.2.8} \]
and covariance function
\[
C_S(x_S, x'_S) = \left( (\Sigma_{\text{sim}}^Z)^{-1} \cdot \text{Cov}_p[J_S(x'_S), Z_{\text{sim}}] \right) \cdot C_J S(x_S, x'_S)
\]

\[
= \left( (\Sigma_{\text{sim}}^Z)^{-1} \cdot \text{Cov}_p[J_S(x'_S), Z_{\text{sim}}] \right) \cdot \text{trace} \left( \text{Cov}_p[J_S(x'_S), Z_{\text{sim}}] \cdot \Sigma_{\text{sim}}^Z \right) - 1
\]

\[
= \left( (\Sigma_{\text{sim}}^Z)^{-1} \cdot \text{Cov}_p[J_S(x'_S), Z_{\text{sim}}] \right) \cdot \text{Cov}_p[J_S(x_S), Z_{\text{sim}}]
\]

(4.2.9)

and variance function

\[
C_P(x_S, x_S) = \left( (\Sigma_{\text{sim}}^Z)^{-1} \cdot \text{Cov}_p[J_S(x_S), Z_{\text{sim}}] \right) \cdot \text{trace} \left( \text{Cov}_p[J_S(x_S), Z_{\text{sim}}] \cdot \Sigma_{\text{sim}}^Z \right) - 1
\]

(4.2.10)

To derive a more explicit expression for (4.2.8)–(4.2.10), note that \(\text{Cov}_p[J_S(x_S), Z_{\text{sim}}]\) is a \(1 \times n\) vector with the \(i\)th element

\[
\text{Cov}_p[J_S(x_S), Z_{\text{sim}}]_i = \text{Cov}_p \left[ \int Y(x)dx_{-S}, Z_{\text{sim}}(x_i) \right]
\]

\[
= \int \text{Cov}_p[Y(x), Y(x_i) + \epsilon(x_i)]dx_{-S}
\]

\[
= \int \text{Cov}_p[Y(x), Y(x_i)]dx_{-S} = \frac{1}{\lambda_Y} \int R_Y(x, x_i; \theta)dx_{-S}
\]

\[
= \frac{1}{\lambda_Y} \prod_{k \notin S} \int R_Y(x_k, x_{i,k}; \theta_k)dx_k \prod_{k \in S} R_Y(x_k, x_{i,k}; \theta_k)\]

\[
\text{sgint}(0, 1, x_{i,k}; \theta_k)
\]

\[
= \frac{1}{\lambda_Y} \prod_{k \notin S} \text{sgint}(0, 1, x_{i,k}; \theta_k) \prod_{k \in S} R_Y(x_k, x_{i,k}; \theta_k)
\]

(4.2.11)

The single integral \(\text{sgint}\) was originally derived for the Gaussian and cubic correlation functions in Moon (2010). In Section 4.3, we present these derivations, and also provide new derivations of \(\text{sgint}\) for the Bohman correlation function.

Below the \(1 \times n\) vector of integrals of the elements of \(\text{Cov}_p[J_S(x_S), Z_{\text{sim}}]\) with respect to \(x_S\),

\[
q = \int_{x_S} \text{Cov}_p[J_S(x_S), Z_{\text{sim}}]dx_S
\]

(4.2.12)
is required. Applying (4.2.11), the $i$th element of $q$ is

$$q_i = \int_{x_S} Cov_p[J_S(x_S), Z_{sim}^i] dx_S$$

$$= \frac{1}{\lambda_Y} \prod_{k \not\in S} \text{sgint}(0, 1, x_{i,k}; \theta_k) \prod_{k \in S} \int_0^1 R^Y(x_k, x_{i,k}; \theta_k) dx_k$$

$$= \frac{1}{\lambda_Y} \prod_{k=1}^d \text{sgint}(0, 1, x_{i,k}; \theta_k). \quad (4.2.13)$$

Also the $n \times n$ matrix of integrals of the elements of $Cov_p[J_S(x_S), Z_{sim}]^T Cov_p[J_S(x_S), Z_{sim}]$ with respect to $x_S$, 

$$C \equiv \int_{x_S} Cov_p[J_S(x_S), Z_{sim}]^T Cov_p[J_S(x_S), Z_{sim}] dx_S, \quad (4.2.14)$$

is needed. Inserting (4.2.11) the $(i,j)$th element of $C$ is

$$(C)_{ij} = \int_{x_S} Cov_p(J_S(x_S), Z_{sim}(x_i)) Cov_p(J_S(x_S), Z_{sim}(x_j)) dx_S$$

$$= \frac{1}{\lambda_Y^2} \prod_{k \not\in S} \text{sgint}(0, 1, x_{i,k}; \theta_k) \text{sgint}(0, 1, x_{j,k}; \theta_k) \prod_{k \in S} \int_0^1 R^Y(x_k, x_{i,k}; \theta_k)R^Y(x_k, x_{j,k}; \theta_k) dx_k$$

$$= \frac{1}{\lambda_Y^2} \prod_{k \not\in S} \text{sgint}(0, 1, x_{i,k}; \theta_k) \text{sgint}(0, 1, x_{j,k}; \theta_k) \prod_{k \in S} \text{mxint}(0, 1, x_{i,k}, x_{j,k}; \theta_k). \quad (4.2.15)$$

The single integral $\text{mxint}$ was originally derived for the Gaussian and cubic correlation functions in Moon (2010). In Section 4.3, we present these derivations, and also provide new derivations of $\text{mxint}$ for the Bohman correlation function.

The joint effect variance $V^j_S = Var_g[j_S(X_S)]$ in (1.7.11) is estimated by the posterior predictive mean of $V^j_S$ given the data $z_{sim}$, i.e., $\hat{V}^j_S = E_P[Var_g[J_S(X_S)]|Z_{sim}]$. 

246
By the mean-variance identity, \( \hat{V}_S^j \) is written as

\[
\hat{V}_S^j = E^P \left[ E_g[J_S^2(x_S)|Z_{sim}] \right] - E^P \left[ E_g[J_S(x_S)]^2 | Z_{sim} \right]
\]

\[
= E_g \left[ E^P \left[ J_S^2(x_S)|Z_{sim} \right] \right] - E^P \left[ E_g[J_S(x_S)]^2 | Z_{sim} \right]
\]

\[
= E_g \left[ Var^P[J_S(x_S)|Z_{sim}] + \left\{ E^P \left[ E_g[J_S(x_S)]^2 | Z_{sim} \right] \right\}^2 \right]
\]

\[
- Var^P \left[ E_g[J_S(x_S)|Z_{sim}] \right] - \left\{ E_g \left[ E^P[J_S(x_S)|Z_{sim}] \right] \right\}^2
\]

\[
= E_g \left[ Var^P[J_S(x_S)|Z_{sim}] + \left\{ E^P[J_S(x_S)|Z_{sim}] \right\}^2 \right]
\]

\[
= E_g \left[ Var^P[J_S(x_S)|Z_{sim}] \right] + Var_g \left[ E^P[J_S(x_S)|Z_{sim}] \right] - Var^P \left[ E_g[J_S(x_S)|Z_{sim}] \right]
\]

and hence

\[
\hat{V}_S^j = E_g \left[ C_S^P(x_S, x_S) \right] + Var_g \left[ m_S^P(x_S) \right] - Var^P \left[ E_g[J_S(x_S)] \right] \quad (4.2.16)
\]

using (4.2.10) and (4.2.8). The three terms in (4.2.16) are calculated as follows. The first component of (4.2.16) is

\[
E_g[C_S^P(x_S, x_S)] = \int_{x_S} C_S^P(x_S, x_S) dx_S
\]

\[
= \int_{x_S} C_S^P(x_S, x_S) dx_S
\]

\[
= \int_{x_S} \text{trace} \left[ (\Sigma_{sim}^Z)^{-1} \text{Cov}_p[J_S(x_S), Z_{sim}]^\top \text{Cov}_p[J_S(x_S), Z_{sim}] \right] dx_S \quad (4.2.17)
\]

\[
= \frac{1}{\lambda_y} \prod_{k \notin S} \text{dbint}(0, 1; \theta_k)
\]

\[
- \text{trace} \left[ (\Sigma_{sim}^Z)^{-1} \int_{x_S} \text{Cov}_p[J_S(x_S), Z_{sim}]^\top \text{Cov}_p[J_S(x_S), Z_{sim}] dx_S \right] \quad (4.2.18)
\]

\[
= \frac{1}{\lambda_y} \prod_{k \notin S} \text{dbint}(0, 1; \theta_k) - \text{trace} \left[ (\Sigma_{sim}^Z)^{-1} C \right] \quad (4.2.19)
\]

247
where (4.2.17) follows from (4.2.10), (4.2.18) follows from (4.2.6), and (4.2.19) follows from (4.2.14). To compute the second term in (4.2.16), first let

\[
\text{mint}(\beta) \equiv \int_{\mathcal{X}_S} m_S^J(x_S)dx_S = \int_{\mathcal{X}_S} \sum_{k_1=0}^{n_{k_1}} \cdots \sum_{k_d=0}^{n_{k_d}} \beta_{k_1 \ldots k_d} \prod_{j \notin S} (k_j + 1)^{-1} \prod_{i \in S} x_i^{k_i}dx_S
\]

\[
= \sum_{k_1=0}^{n_{k_1}} \cdots \sum_{k_d=0}^{n_{k_d}} \beta_{k_1 \ldots k_d} \prod_{i=1}^{d} (k_j + 1)^{-1},
\]

\[
\text{msqint}(\beta) \equiv \int_{\mathcal{X}_S} (m_S^J(x_S))^2dx_S
\]

\[
= \int_{\mathcal{X}_S} \left\{ \sum_{k_1,\ldots,k_d \in k_1', \ldots, k_d'} \beta_{k_1 \ldots k_d} \beta_{k_1' \ldots k_d'} \prod_{j \notin S} (k_j + 1) \prod_{j \notin S} (k_j' + 1) \prod_{i \in S} x_i^{k_i+k_i'} \right\} dx_S
\]

\[
= \sum_{k_1,\ldots,k_d \in k_1', \ldots, k_d'} \beta_{k_1 \ldots k_d} \beta_{k_1' \ldots k_d'} \prod_{j \notin S} (k_j + 1) \prod_{j \notin S} (k_j' + 1) \prod_{i \in S} x_i^{k_i+k_i'} \int_{0}^{1} x_i^{k_i+k_i'}dx_i
\]

\[
= \sum_{k_1,\ldots,k_d \in k_1', \ldots, k_d'} \beta_{k_1 \ldots k_d} \beta_{k_1' \ldots k_d'} \prod_{j \notin S} (k_j + 1) \prod_{j \notin S} (k_j' + 1) \prod_{i \in S} (k_i + k_i' + 1)^{-1},
\]

and

\[
v^\top = \int_{\mathcal{X}_S} m_S^J(x_S) \text{Cov}_p[J_S(X_S), Z_{sim}]dx_S,
\]

which is an \(1 \times n\) vector whose \(h^{th}\) element is

\[
v_h = \int_{\mathcal{X}_S} m_S^J(x_S) \text{Cov}_p[J_S(X_S), Z_{sim}(x_h)]dx_S
\]

\[
= \frac{1}{\lambda_Y} \int_{\mathcal{X}_S} \left[ \sum_{k_1=0}^{n_{k_1}} \cdots \sum_{k_d=0}^{n_{k_d}} \beta_{k_1 \ldots k_d} \prod_{j \notin S} (k_j + 1)^{-1} \prod_{i \in S} x_i^{k_i} \prod_{i \in S} R^Y (x_i, x_{h,i}, ; \theta_i) \right] dx_S
\]

\[
\times \prod_{j \notin S} \text{sgint} (0, 1, x_{h,j}, ; \theta_j)
\]

\[
= \frac{1}{\lambda_Y} \prod_{j \notin S} \text{sgint} (0, 1, x_{h,j}, ; \theta_j)
\]

248
\[
\times \sum_{k_1=0}^{n_{k_1}} \ldots \sum_{k_d=0}^{n_{k_d}} \left\{ \beta_{k_1 \ldots k_d} \prod_{j \notin S} (k_j + 1)^{-1} \prod_{i \in S} \int_0^1 x_i^{k_i} R^y (x_i, x_{h,i}; \theta_i) \, dx_i \right\} \\
= \left[ \frac{1}{\lambda_Y} \prod_{j \notin S} \text{sgint} (0, 1, x_{h,j}; \theta_j) \right] \\
\times \sum_{k_1=0}^{n_{k_1}} \ldots \sum_{k_d=0}^{n_{k_d}} \left\{ \beta_{k_1 \ldots k_d} \prod_{j \notin S} (k_j + 1)^{-1} \prod_{i \in S} \text{prodint}(0, 1, x_{h,i}; k_i, \theta_i) \right\},
\]

where the integral \text{prodint} is derived in Section 4.3 for the Gaussian, Bohman and cubic correlation functions. Because

\[
E_g \left[ (m_p^S (X_S))^2 \right] = \int_{X_S} (m_p^S (x_S))^2 \, dx_S \\
+ 2 \int_{X_S} m_p^S (x_S) \text{Cov}_p [J_S (X_S), Z_{sim}] d x_S (\Sigma_{sim}^{Z_{sim}})^{-1} (z_{sim} - m_{sim}^{Z_{sim}}) \\
+ (z_{sim} - m_{sim}^{Z_{sim}})^\top (\Sigma_{sim}^{Z_{sim}})^{-1} Q (\Sigma_{sim}^{Z_{sim}})^{-1} C (\Sigma_{sim}^{Z_{sim}})^{-1} (z_{sim} - m_{sim}^{Z_{sim}}),
\]

and

\[
\left( E_g \left[ (m_p^S (X_S))^2 \right] \right)^2 = \left\{ \int_{X_S} m_p^S (x_S) \, dx_S \right\}^2 \\
+ 2 \int_{X_S} m_p^S (x_S) \, dx_S \, Q^\top (\Sigma_{sim}^{Z_{sim}})^{-1} (z_{sim} - m_{sim}^{Z_{sim}}) \\
+ (z_{sim} - m_{sim}^{Z_{sim}})^\top (\Sigma_{sim}^{Z_{sim}})^{-1} Q Q^\top (\Sigma_{sim}^{Z_{sim}})^{-1} (z_{sim} - m_{sim}^{Z_{sim}}),
\]

the second term in (4.2.16) can be expressed as

\[
\text{Var}_g [m_p^S (X_S)] = E_g \left[ (m_p^S (X_S))^2 \right] - (E_g \left[ m_p^S (X_S) \right])^2 \\
= \int_{X_S} (m_p^S (x_S))^2 \, dx_S - \left\{ \int_{X_S} m_p^S (x_S) \, dx_S \right\}^2
\]

249
\[ +2 \int m_S^T(x_S) \text{Cov}_P[J_S(X_S), Z_{\text{sim}}] d x_S (\Sigma_{\text{sim}}^Z)^{-1} (z_{\text{sim}} - m_{\text{sim}}^Z) \]
\[- \int m_S^T(x_S) d x_S q^\top (\Sigma_{\text{sim}}^Z)^{-1} (z_{\text{sim}} - m_{\text{sim}}^Z) \]
\[+ (z_{\text{sim}} - m_{\text{sim}}^Z)^\top (\Sigma_{\text{sim}}^Z)^{-1} (C - qq^\top) (\Sigma_{\text{sim}}^Z)^{-1} (z_{\text{sim}} - m_{\text{sim}}^Z) = \text{msqint}(\beta) - \text{mint}^2(\beta) \]
\[+ 2 (v^\top - \text{mint}(\beta) q^\top) (\Sigma_{\text{sim}}^Z)^{-1} (z_{\text{sim}} - m_{\text{sim}}^Z) \]
\[+ (z_{\text{sim}} - m_{\text{sim}}^Z)^\top (\Sigma_{\text{sim}}^Z)^{-1} (C - qq^\top) (\Sigma_{\text{sim}}^Z)^{-1} (z_{\text{sim}} - m_{\text{sim}}^Z) . \]

The third term of (4.2.16) is

\[ \text{Var}^P[E_g[J_S(X_S)]|Z_{\text{sim}}] = \text{Cov}^P[E_g[J_S(X_S)], E_g[J_S(X_S)]|Z_{\text{sim}}] \]
\[= \int \int C_{S}^P(x_S, x_S') d x_S d x_S' \]
\[= \int \int C_{S}^J(x_S, x_S') d x_S d x_S' \]
\[ - \int \int trace \left[ (\Sigma_{\text{sim}}^Z)^{-1} \text{Cov}_P[J_S(x_S'), Z_{\text{sim}}] \text{Cov}_P[J_S(x_S), Z_{\text{sim}}] \right] d x_S d x_S' \quad (4.2.21) \]
\[= \frac{1}{\lambda_Y} \prod_{k \in S} \text{dbint}(0, 1; \theta_k) \prod_{k \in S} \int_0^1 \int_0^1 R^Y(x_k, x'_k; \theta_k) d x_k d x'_k \quad \text{dbint}(0, 1; \theta_k) \]
\[ - \text{trace} \left[ (\Sigma_{\text{sim}}^Z)^{-1} \int \text{Cov}_P[J_S(x_S'), Z_{\text{sim}}] d x_S' \int \text{Cov}_P[J_S(x_S), Z_{\text{sim}}] d x_S \right] \quad (4.2.22) \]
\[= \frac{1}{\lambda_Y} \prod_{k=1}^d \text{dbint}(0, 1; \theta_k) - \text{trace} \left[ (\Sigma_{\text{sim}}^Z)^{-1} qq^\top \right] \quad (4.2.23) \]
where (4.2.21) follows from (4.2.9), (4.2.22) follows from (4.2.5), and (4.2.23) follows from (4.2.12). In sum, \( \hat{V}_S^j \) in (4.2.16) can be expressed as

\[
\hat{V}_S^j = \left\{ \frac{1}{\lambda_Y} \prod_{k \notin S} \text{dbint}(0, 1; \theta_k) - \text{trace} \left[ (\Sigma_{\text{sim}}^Z)^{-1} C \right] \right\} 
+ \left\{ \text{msqint}(\beta) - \text{mint}^2(\beta) + 2 \left( \nu^\top - \text{mint}(\beta) q^\top \right) (\Sigma_{\text{sim}}^Z)^{-1} (z_{\text{sim}} - m_{\text{sim}}^Z) 
+ (z_{\text{sim}} - m_{\text{sim}}^Z)^\top (\Sigma_{\text{sim}}^Z)^{-1} (C - qq^\top) (\Sigma_{\text{sim}}^Z)^{-1} (z_{\text{sim}} - m_{\text{sim}}^Z) \right\} 
- \left\{ \frac{1}{\lambda_Y} \prod_{k=1}^d \text{dbint}(0, 1; \theta_k) - \text{trace} \left[ (\Sigma_{\text{sim}}^Z)^{-1} q q^\top \right] \right\} 
\]

(4.2.24)

The estimate \( \hat{V} \) of the total variance \( V \) can be expressed as (4.2.24) when \( S = \{1, \ldots, d\} \). The main effect sensitivity index for the individual input \( x_i \) is estimated by

\[
\hat{S}_i = \frac{\hat{V}_i^j}{\hat{V}} \quad (4.2.25)
\]

where \( S = \{i\} \), and the total effect sensitivity index is estimated by

\[
\hat{T}_i = \frac{\hat{V}_i^j - \hat{V}_{i-1}^j}{\hat{V}} \quad (4.2.26)
\]

where \( \hat{V}_{i-1}^j \) is obtained from (4.2.24) when \( S = \{1, \ldots, i-1, i+1, \ldots, d\} \).

We will briefly compare the functional form of \( \hat{V}_S^j \) when one assumes a constant mean as opposed to a polynomial mean. With a constant mean, the first and third components of (4.2.16) are unchanged, but the second component simplifies to

\[
\begin{align*}
\text{Var}_g[m_{\text{sim}}^p(X_S)] &= E_g[(m_{\text{sim}}^p(X_S) - E_g[m_{\text{sim}}^p(X_S)])^2] \\
&= E_g \left[ ((\text{Cov}_p[J_S(X_S), Z_{\text{sim}}] - q^\top) (\Sigma_{\text{sim}}^Z)^{-1} (z_{\text{sim}} - m_{\text{sim}}^Z))^2 \right] \\
&= (z_{\text{sim}} - m_{\text{sim}}^Z)^\top (\Sigma_{\text{sim}}^Z)^{-1} \\
&\times E_g \left[ ((\text{Cov}_p[J_S(X_S), Z_{\text{sim}}] - q^\top)^\top (\text{Cov}_p[J_S(X_S), Z_{\text{sim}}] - q^\top) \right] \\
&\times (\Sigma_{\text{sim}}^Z)^{-1} (z_{\text{sim}} - m_{\text{sim}}^Z) \\
&= (z_{\text{sim}} - m_{\text{sim}}^Z)^\top (\Sigma_{\text{sim}}^Z)^{-1} (C - qq^\top) (\Sigma_{\text{sim}}^Z)^{-1} (z_{\text{sim}} - m_{\text{sim}}^Z) 
\end{align*}
\]

251
so that

\[
\hat{V}_j^S = \left\{ \frac{1}{\lambda_Y} \prod_{k \not\in S} \text{dbint}(0, 1; \theta_k) - \text{trace} \left[ \left( \Sigma_{\text{sim}}^Z \right)^{-1} C \right] \right\} + \left\{ (z_{\text{sim}} - m_{\text{sim}}^Z)^\top \left( \Sigma_{\text{sim}}^Z \right)^{-1} (C - qq^\top) \left( \Sigma_{\text{sim}}^Z \right)^{-1} (z_{\text{sim}} - m_{\text{sim}}^Z) \right\}
\]

Therefore, \( \hat{V}_j^S \) is considerably simpler to compute, as it still depends on \( \text{sgint}, \text{dbint}, \) and \( \text{mxint} \), but no longer depends on \( \text{prodint} \).

### 4.3 Computing the Integrals for Three Correlation Functions

Analytical forms of \( \text{sgint}, \text{dbint}, \text{mxint}, \) and \( \text{prodint} \) are derived for the Gaussian, Bohman, and cubic correlation functions. Let \( R^Y(x, \eta; \theta) \) denote one of these correlation functions with correlation parameter \( \theta > 0 \) and two input values \( x, \eta \in [l, u] \).

Then the four integrals are

\[
\begin{align*}
\text{sgint}(l, u, \eta; \theta) & \equiv \frac{1}{u - l} \int_l^u R^Y(x, \eta; \theta) \, dx \\
\text{dbint}(l, u; \theta) & \equiv \frac{1}{(u - l)^2} \int_l^u \int_l^u R^Y(x, \eta; \theta) \, d\eta \, dx \\
\text{mxint}(l, u, \eta_1, \eta_2; \theta) & \equiv \frac{1}{u - l} \int_l^u R^Y(x, \eta_1; \theta) R^Y(x, \eta_2; \theta) \, dx \\
\text{prodint}(l, u, \eta; k, \theta) & \equiv \frac{1}{u - l} \int_l^u x^k R^Y(x, \eta; \theta) \, dx, 
\end{align*}
\]

where \( k \in \{0, 1, 2, \ldots\} \). Out derivations of \( \text{sgint}, \text{dbint}, \) and \( \text{mxint} \) will be slightly more general than required in Section 4.2, as we will allow \( u \) and \( l \) to be any constants that satisfy \( u > l \). For \( \text{prodint} \), we will restrict our calculations to the \( [l, u] = [0, 1] \) case.
4.3.1 Gaussian Correlation Function

The Gaussian correlation function is

\[ R^Y(x, \eta; \theta) = \exp \left[ -\theta(x - \eta)^2 \right] . \]

Moon (2010) shows that the closed forms of \texttt{sgint}, \texttt{dbint}, and \texttt{mxint} are

\[
\texttt{sgint}(l, u, \eta; \theta) = \frac{1}{u - l} \int_l^u \exp[-\theta(x - \eta)^2]dx
= \frac{1}{u - l} \sqrt{\frac{\pi}{\theta}} \left[ \Phi \left( (u - \eta)\sqrt{2\theta} \right) - \Phi \left( (l - \eta)\sqrt{2\theta} \right) \right]
\]

where \( \Phi(\cdot) \) denotes a cumulative distribution function (cdf) of the standard normal distribution,

\[
\texttt{dbint}(l, u; \theta) = \frac{1}{(u - l)^2} \int_l^u \int_l^u \exp[-\theta(x - \eta)^2]d\eta dx
= \frac{1}{(u - l)^2} \left\{ \frac{1}{\theta} \left[ \sqrt{2\pi}\phi \left( (u - l)\sqrt{2\theta} \right) - 1 \right] + (u - l)\sqrt{\frac{\pi}{\theta}} \left[ 2\Phi \left( (u - l)\sqrt{2\theta} \right) - 1 \right] \right\}
\]

where \( \phi(\cdot) \) denotes a probability density function (pdf) of the standard normal distribution, and

\[
\texttt{mxint}(l, u, \eta_1, \eta_2; \theta) = \frac{1}{u - l} \int_l^u \exp[-\theta(x - \eta_1)^2] \exp[-\theta(x - \eta_2)^2]dx
= \exp \left[ -\frac{1}{2}\theta(\eta_1 - \eta_2)^2 \right] \texttt{sgint} \left( l, u, \frac{\eta_1 + \eta_2}{2}; 2\theta \right).
\]

To derive an analytic expression for \texttt{prodint} when \( l = 0 \) and \( u = 1 \), we will rely on the following result for finding moments of truncated normal random variables from Dhrymes (2005).

\textbf{Result 1.} Suppose that \( U \sim N(\mu, \sigma^2) \). For any constant \( c \), any integers \( m \geq r \geq 0 \) and \( h = \frac{c - \mu}{\sigma} \), the integral

\[
I_r^h \equiv \frac{1}{\Phi(h)} \int_{-\infty}^{h} \xi^r \phi(\xi)d\xi
\]

253
can be expressed recursively as

\[
I^h_0 = 1 \\
I^h_1 = -\frac{\phi(h)}{\Phi(h)} \\
I^h_r = -h^{r-1}\frac{\phi(h)}{\Phi(h)} + (r-1)I^h_{r-2} \quad (r \in \{2, 3, 4, \ldots\}).
\]

We can then express the conditional moments of \([U|U \leq c]\) as

\[
E[U^m|U \leq c] = \sum_{r=0}^{m} \binom{m}{r} \mu^{m-r}\sigma^r I^h_r.
\]

(4.3.5)

Proof. We first obtain the recursive expression for \(I^h_r\).

\[
I^h_0 = \frac{1}{\Phi(h)} \int_{-\infty}^{h} \phi(\xi)d\xi = 1,
\]

\[
I^h_1 = \frac{1}{\Phi(h)} \int_{-\infty}^{h} \xi\phi(\xi)d\xi = \left. \frac{-\phi(\xi)}{\Phi(h)} \right|_{-\infty}^{h} = \frac{-\phi(h)}{\Phi(h)},
\]

and for \(r \in \{2, 3, 4, \ldots\}\), we can use integration by parts with

\[
\begin{aligned}
s &= \xi^{r-1} \\
dt &= \xi\phi(\xi) \\
ds &= (r-1)\xi^{r-2} \\
t &= -\phi(\xi)
\end{aligned}
\]

to obtain

\[
I^h_r = \frac{1}{\Phi(h)} \int_{-\infty}^{h} \xi^r\phi(\xi)d\xi
\]

\[
= \frac{1}{\Phi(h)} \left[ -h^{r-1}\phi(h) + (r-1) \int_{-\infty}^{h} \xi^{r-2}\phi(\xi)d\xi \right]
\]

\[
= \frac{1}{\Phi(h)} \left[ -h^{r-1}\phi(h) + (r-1)I^h_{r-2} \right]
\]

Now, we will simplify \(E[U^m|U \leq c]\). The probability density function of \([U|U \leq c]\) is

\[
f(u|u \leq c) = \frac{\frac{\phi(u-\mu)}{\sigma}}{\Phi\left(\frac{c-\mu}{\sigma}\right)}1_{[u \leq c]}.
\]
We then have
\[
E [U^m | U \leq c] = \frac{1}{\Phi(h)} \int_{-\infty}^{c} u^m \frac{1}{\sigma} \phi \left( \frac{u - \mu}{\sigma} \right) du.
\]
Using the change of variable \( \xi = \frac{u - \mu}{\sigma} \) and the binomial theorem, we can conclude that
\[
E [U^m | U \leq c] = \frac{1}{\Phi(h)} \int_{-\infty}^{h} (\sigma \xi + \mu)^m \phi(\xi) d\xi
\]
In the following, we will derive the closed form expressions for the four integrals (4.3.1)-(4.3.4).

1) sgint

Letting \( l^* = \max(l, \eta - \theta) \) and \( u^* = \min(u, \eta + \theta) \), we have

\[
\text{sgint}(l, u, \eta; \theta) = \frac{1}{u - l} \int_l^u R_Y(x, \eta; \theta) \, dx
\]

\[
= \frac{1}{u - l} \int_l^u \left\{ \left( 1 - \frac{\eta - x}{\theta} \right) \cos \left( \frac{\pi (\eta - x)}{\theta} \right) + \frac{1}{\pi} \sin \left( \frac{\pi (\eta - x)}{\theta} \right) \right\} \, dx
\]

\[
+ \frac{1}{u - l} \int_u^u \left\{ \left( 1 - \frac{x - \eta}{\theta} \right) \cos \left( \frac{\pi (x - \eta)}{\theta} \right) + \frac{1}{\pi} \sin \left( \frac{\pi (x - \eta)}{\theta} \right) \right\} \, dx.
\]

Making the change of variables \( s = \frac{\eta - x}{\theta} \) and \( t = \frac{x - \eta}{\theta} \), and letting

\[
l^{**}(\eta) = \frac{\eta - l^*}{\theta \pi} = \frac{\eta - \max(l, \eta - \theta)}{\theta \pi} = \min \left( \frac{\eta - l}{\theta \pi}, \pi \right)
\]

\[
u^{**}(\eta) = \frac{u^* - \eta}{\theta \pi} = \min \left( \frac{u + \theta - \eta}{\theta \pi}, \frac{\theta}{\theta \pi}, \pi \right),
\]

we have

\[
\text{sgint}(l, u, \eta; \theta) = \frac{1}{u - l} \left\{ \int_0^{l^{**}(\eta)} \left\{ \left( 1 - \frac{t}{\pi} \right) \cos(t) + \frac{1}{\pi} \sin(t) \right\} \frac{\theta}{\pi} \, dt
\]

\[- \int_{l^{**}(\eta)}^0 \left\{ \left( 1 - \frac{s}{\pi} \right) \cos(s) + \frac{1}{\pi} \sin(s) \right\} \frac{\theta}{\pi} \, ds \right\}
\]

\[
= \frac{1}{u - l} \left\{ \frac{4\theta}{\pi^2} - \frac{2\theta}{\pi^2} \cos(l^{**}(\eta)) - \frac{2\theta}{\pi^2} \cos(u^{**}(\eta))
\]

\[+ \left\{ \left( \frac{\theta}{\pi} - \frac{\theta l^{**}(\eta)}{\pi^2} \right) \sin(l^{**}(\eta)) + \left( \frac{\theta}{\pi} - \frac{\theta u^{**}(\eta)}{\pi^2} \right) \sin(u^{**}(\eta)) \right\} \right\}
\]

2) dbint

Letting \( a = \max(l, u - \theta) \), \( b = \min(u, \theta) \), \( l^{**}(x) = \min \left( \frac{x - l}{\theta \pi}, \pi \right) \), and \( u^{**}(x) = \frac{u - \eta}{\theta \pi} \),
\[
\min \left( \left( \frac{u-x}{\theta} \right) \pi, \pi \right), \text{ we have}
\]

\[
dbint(l, u; \theta) = \frac{1}{(u-l)^2} \int_l^u \int_l^u R^Y(x, \eta; \theta) d\eta dx
\]

\[
= \frac{1}{u-l} \int_0^1 \text{sgint}(x; \theta) dx
\]

\[
= \frac{1}{(u-l)^2} \int_l^u \left\{ \frac{4\theta}{\pi^2} - \frac{2\theta}{\pi^2} \cos(u^*(x)) - \frac{2\theta}{\pi^2} \cos(l^*(x)) + \left( \frac{\theta}{\pi} - \frac{\theta u^*(x)}{\pi^2} \right) \sin(u^*(x)) + \left( \frac{\theta}{\pi} - \frac{\theta l^*(x)}{\pi^2} \right) \sin(l^*(x)) \right\} dx
\]

\[
= \frac{4\theta}{(u-l)^2} + \frac{1}{(u-l)^2} \left\{ \int_l^u \frac{2\theta}{\pi^2} dx + \int_l^a \frac{2\theta}{\pi^2} dx \right\}
\]

\[
+ \frac{1}{(u-l)^2} \int_l^b \left\{ \frac{\theta}{\pi} \left( 1 - \frac{x-l}{\theta} \right) \sin \left( \frac{x-l}{\theta} \right) - \frac{2\theta}{\pi^2} \cos \left( \frac{x-l}{\theta} \right) \right\} dx
\]

\[
+ \frac{1}{(u-l)^2} \int_a^u \left\{ \frac{\theta}{\pi} \left( 1 - \frac{u-x}{\theta} \right) \sin \left( \frac{u-x}{\theta} \right) - \frac{2\theta}{\pi^2} \cos \left( \frac{u-x}{\theta} \right) \right\} dx
\]

Simplifying some terms, making the change of variables \( s = \frac{x-l}{\theta} \pi \) and \( t = \frac{u-x}{\theta} \pi \), and letting \( a^* = \pi \frac{u-a}{\theta} \) and \( b^* = \pi \frac{b-l}{\theta} \), we then have

\[
\dbint(l, u; \theta) = \frac{4\theta}{(u-l)^2} - \frac{2\theta}{(u-l)^2 \pi^2} (l + b - a)
\]

\[
+ \frac{1}{(u-l)^2} \int_0^{b^*} \left\{ \left( 1 - \frac{\theta}{\pi} \right) \sin(s) - \frac{2\theta}{\pi} \cos(s) \right\} \frac{\theta^2}{\pi^2} ds
\]

\[
- \frac{1}{(u-l)^2} \int_0^{a^*} \left\{ \left( 1 - \frac{\theta}{\pi} \right) \sin(t) - \frac{2\theta}{\pi} \cos(t) \right\} \frac{\theta^2}{\pi^2} dt
\]

Now, notice that \( a = l, b = u, a^* = b^* = \pi \frac{(u-l)}{\theta} \), and

\[
\dbint(l, u; \theta) = \frac{4\theta}{(u-l)^2} + \frac{1}{(u-l)^2} \int_0^{\pi (u-l)/\theta} \left\{ \left( 1 - \frac{\theta}{\pi} \right) \sin(s) - \frac{2\theta}{\pi} \cos(s) \right\} \frac{\theta^2}{\pi^2} ds
\]

\[
+ \frac{1}{(u-l)^2} \int_{\pi (u-l)/\theta}^\pi \left\{ \left( 1 - \frac{\theta}{\pi} \right) \sin(t) - \frac{2\theta}{\pi} \cos(t) \right\} \frac{\theta^2}{\pi^2} dt.
\]

257
if and only if $\theta \geq u - l$. Similarly, $a = u - \theta$, $b = l + \theta$, $a^* = b^* = \pi$ and

$$
\text{dbint}(l, u; \theta) = \frac{4\theta}{(u - l)^2\pi^2} + \frac{4\theta}{(u - l)^2\pi^2} (\theta + l - u)
+ \frac{1}{(u - l)^2} \int_0^\pi \left\{ \left( 1 - \frac{s}{\pi} \right) \sin(s) - \frac{2}{\pi} \cos(s) \right\} \frac{\theta^2}{\pi^2} ds
+ \frac{1}{(u - l)^2} \int_0^\pi \left\{ \left( 1 - \frac{t}{\pi} \right) \sin(t) - \frac{2}{\pi} \cos(t) \right\} \frac{\theta^2}{\pi^2} dt.
$$

if and only if $\theta < u - l$. Therefore, we can express $\text{dbint}(\theta)$ as the piecewise function

$$
\text{dbint}(l, u; \theta) = \begin{cases} 
\frac{4\theta}{(u - l)^2\pi^2} + \frac{2\theta^2}{(u - l)^2\pi^2} (2\theta - 2u + 2l), & \theta < u - l; \\
\frac{4\theta}{(u - l)^2\pi^2} + \frac{2\theta^2}{(u - l)^2\pi^2} \left\{ 1 + \left( \frac{u - l}{\theta} - 1 \right) \cos\left( \frac{\pi(u - l)}{\theta} \right) - \frac{3}{\pi} \sin\left( \frac{\pi(u - l)}{\theta} \right) \right\}, & \theta \geq u - l.
\end{cases}
$$

3) mxint

Without loss of generality, assume that $\eta_1 < \eta_2$. Also, let $l^* = \max(l, \eta - \theta)$ and $u^* = \min(u, \eta + \theta)$. To calculate $\text{mxint}(l, u, \eta_1, \eta_2; \theta)$, we must first identify the regions of $[l, u]$ for which $R_Y(x, \eta_1; \theta)R_Y(x, \eta_2; \theta) \neq 0$. These regions will depend on the relationship between $|\eta_1 - \eta_2|$ and $\theta$.

• Case 1: $|\eta_1 - \eta_2| \geq 2\theta$

Suppose that $|x - \eta_1| < \theta$ so that $R_Y(x, \eta_1; \theta) > 0$. Then, by the so-called reverse triangle inequality,

$$
|x - \eta_2| = |x - \eta_1 - (\eta_2 - \eta_1)|
\geq |x - \eta_1| - |\eta_2 - \eta_1|
\geq \theta,
$$

so $R_Y(x, \eta_2; \theta) = 0$. Similarly, if $|x - \eta_2| \leq \theta$ so that $R_Y(x, \eta_2; \theta) > 0$, then $|x - \eta_1| > \theta$ and $R_Y(x, \eta_1; \theta) = 0$. Therefore, $R_Y(x, \eta_1; \theta)$ and $R_Y(x, \eta_2; \theta)$ cannot simultaneously be nonzero, so $R_Y(x, \eta_1; \theta)R_Y(x, \eta_2; \theta) = 0$ and $\text{mxint}(\eta_1, \eta_2; \theta) = 0$. 

258
• Case 2: $\theta \leq |\eta_1 - \eta_2| < 2\theta$

If $x < \eta_1$, then $|x - \eta_2| > |\eta_1 - \eta_2| \geq \theta$, so $R^Y(x, \eta_2; \theta) = 0$. Similarly, $x > \eta_2$ implies that $R^Y(x, \eta_1; \theta) = 0$. Therefore, $R^Y(x, \eta_1; \theta) R^Y(x, \eta_2; \theta) = 0$ if $x \notin [\eta_1, \eta_2]$. Moreover, for $x \in [\eta_1, \eta_2]$, $R^Y(x, \eta_1; \theta) > 0$ if and only if $x < \eta_1 + \theta$, and $R^Y(x, \eta_2; \theta) > 0$ if and only if $x > \eta_2 - \theta$. Thus, we have

$$\text{mxint}(l, u, \eta_1, \eta_2; \theta) = \frac{1}{u - l} \int_{\eta_2 - \theta}^{\eta_1 + \theta} R^Y(x, \eta_1; \theta) R^Y(x, \eta_2; \theta) dx$$

$$= \frac{1}{u - l} \int_{\eta_2 - \theta}^{\eta_1 + \theta} \left\{ \left( 1 - \frac{x - \eta_1}{\theta} \right) \cos \left( \frac{\pi(x - \eta_1)}{\theta} \right) + \frac{1}{\pi} \sin \left( \frac{\pi(x - \eta_1)}{\theta} \right) \right\} \times \left\{ \left( 1 - \frac{(\eta_2 - x)}{\theta} \right) \cos \left( \frac{\pi(\eta_2 - x)}{\theta} \right) + \frac{1}{\pi} \sin \left( \frac{\pi(\eta_2 - x)}{\theta} \right) \right\} dx$$

• Case 3: $|\eta_1 - \eta_2| < \theta$

First, notice that for $x \in (\eta_1, \eta_2)$, $|x - \eta_1| < |\eta_1 - \eta_2| < \theta$ and $|x - \eta_2| < |\eta_1 - \eta_2| < \theta$, so $R^Y(x, \eta_1; \theta) > 0$ and $R^Y(x, \eta_2; \theta) > 0$. Additionally, for $x < \eta_1$, $R^Y(x, \eta_1; \theta)$ and $R^Y(x, \eta_2; \theta)$ are simultaneously nonzero if and only if $x > \eta_1 - \theta$, i.e, $\eta_1 - \theta < x < \eta_1$, and for $x > \eta_2$, $R^Y(x, \eta_1; \theta)$ and $R^Y(x, \eta_2; \theta)$ are simultaneously nonzero if and only if $x < \eta_2 + \theta$, i.e, $\eta_2 < x < \eta_2 + \theta$. Therefore, $R^Y(x, \eta_1; \theta) R^Y(x, \eta_2; \theta) > 0$ if and only if $\eta_1 - \theta < x < \eta_2 + \theta$. Since the bounds on $\text{mxint}(\eta_1, \eta_2; \theta)$ are 0 and 1, if we let $l^* = \max(\eta_1 - \theta, l)$ and $u^* = \min(\eta_2 + \theta, u)$, we then have

$$\text{mxint}(l, u, \eta_1, \eta_2; \theta)$$

$$= \int_{l^*}^{\eta_1} R^Y(x, \eta_1; \theta) R^Y(x, \eta_2; \theta) dx$$

$$+ \int_{\eta_1}^{\eta_2} R^Y(x, \eta_1; \theta) R^Y(x, \eta_2; \theta) dx$$

$$+ \int_{\eta_2}^{u^*} R^Y(x, \eta_1; \theta) R^Y(x, \eta_2; \theta) dx$$
= \frac{1}{u-l} \int_{l^*}^{m_1} \left\{ \left(1 - \frac{\eta_1 - x}{\theta} \right) \cos \left( \frac{\pi(\eta_1 - x)}{\theta} \right) + \frac{1}{\pi} \sin \left( \frac{\pi(\eta_1 - x)}{\theta} \right) \right\} dx \\
\times \left\{ \left(1 - \frac{\eta_2 - x}{\theta} \right) \cos \left( \frac{\pi(\eta_2 - x)}{\theta} \right) + \frac{1}{\pi} \sin \left( \frac{\pi(\eta_2 - x)}{\theta} \right) \right\} \\
+ \frac{1}{u-l} \int_{m_1}^{m_2} \left\{ \left(1 - \frac{x - \eta_1}{\theta} \right) \cos \left( \frac{\pi(x - \eta_1)}{\theta} \right) + \frac{1}{\pi} \sin \left( \frac{\pi(x - \eta_1)}{\theta} \right) \right\} dx \\
\times \left\{ \left(1 - \frac{x - \eta_2}{\theta} \right) \cos \left( \frac{\pi(x - \eta_2)}{\theta} \right) + \frac{1}{\pi} \sin \left( \frac{\pi(x - \eta_2)}{\theta} \right) \right\} dx \\
+ \frac{1}{u-l} \int_{m_2}^{u^*} \left\{ \left(1 - \frac{x - \eta_1}{\theta} \right) \cos \left( \frac{\pi(x - \eta_1)}{\theta} \right) + \frac{1}{\pi} \sin \left( \frac{\pi(x - \eta_1)}{\theta} \right) \right\} dx \\
\times \left\{ \left(1 - \frac{x - \eta_2}{\theta} \right) \cos \left( \frac{\pi(x - \eta_2)}{\theta} \right) + \frac{1}{\pi} \sin \left( \frac{\pi(x - \eta_2)}{\theta} \right) \right\} dx \\
(4.3.8)

We can combine all three cases into one compact expression. First, let \( m_l = \text{med}(\eta_1, l, \eta_2 - \theta) \) and \( m_u = \text{med}(\eta_2, u, \eta_1 + \theta) \). Then, we have

\[
\text{mxint}(l, u, \eta_1, \eta_2; \theta) = \frac{1}{[\eta_1 - \eta_2 < 2\theta]} \frac{1}{u-l} \left\{ \begin{array}{l}
\int_{m_1}^{m_2} \left(1 - \frac{\eta_1 - x}{\theta} \right) \cos \left( \frac{\pi(\eta_1 - x)}{\theta} \right) + \frac{1}{\pi} \sin \left( \frac{\pi(\eta_1 - x)}{\theta} \right) \right\} dx \\
+ \frac{1}{u-l} \int_{m_1}^{m_2} \left(1 - \frac{x - \eta_1}{\theta} \right) \cos \left( \frac{\pi(x - \eta_1)}{\theta} \right) + \frac{1}{\pi} \sin \left( \frac{\pi(x - \eta_1)}{\theta} \right) \right\} dx \\
+ \frac{1}{u-l} \int_{m_2}^{u^*} \left(1 - \frac{x - \eta_1}{\theta} \right) \cos \left( \frac{\pi(x - \eta_1)}{\theta} \right) + \frac{1}{\pi} \sin \left( \frac{\pi(x - \eta_1)}{\theta} \right) \right\} dx \\
\end{array} \right. \right)}
\]

By expanding the preceding integrands using the angle sum and difference identities

\[
\sin(\alpha \pm \beta) = \sin(\alpha) \cos(\beta) \pm \cos(\alpha) \sin(\beta) \quad (4.3.9)
\]

\[
\cos(\alpha \pm \beta) = \cos(\alpha) \cos(\beta) \mp \sin(\alpha) \sin(\beta) \quad (4.3.10)
\]

260
as well as the product to sum identities

\[
\begin{align*}
\cos(\alpha) \cos(\beta) &= \frac{\cos(\alpha - \beta) + \cos(\alpha + \beta)}{2} \\
\sin(\alpha) \sin(\beta) &= \frac{\cos(\alpha - \beta) - \cos(\alpha + \beta)}{2} \\
\sin(\alpha) \cos(\beta) &= \frac{\sin(\alpha + \beta) + \sin(\alpha - \beta)}{2} \\
\cos(\alpha) \sin(\beta) &= \frac{\sin(\alpha + \beta) - \sin(\alpha - \beta)}{2},
\end{align*}
\]

\textbf{mxint} can be expressed as the sum of integrals of the form

\[
\begin{align*}
\int c_1 x^k \sin(c_2 x) dx \\
\int c_1 x^k \cos(c_2 x) dx,
\end{align*}
\]

where \( k \) is a non-negative integer, and \( c_1 \) and \( c_2 \) are constants. Analytic expressions of these integrals can be found using the recursive formulas

\[
\begin{align*}
\int c_1 x^k \sin(c_2 x) dx &= -c_1 \frac{x^k}{c_2} \cos(c_2 x) + c_1 \frac{k}{c_2} \int x^{k-1} \cos(c_2 x) dx \\
\int c_1 x^k \cos(c_2 x) dx &= c_1 \frac{x^k}{c_2} \sin(c_2 x) - c_1 \frac{k}{c_2} \int x^{k-1} \sin(c_2 x) dx.
\end{align*}
\]

One \textit{could} attempt to simplify (4.3.8) by expanding all integrands to products of polynomial and trigonometric functions and applying the recursive formula above. However, we have chosen to relegate the rather tedious task of calculating the closed form expression of (4.3.8) to MATLAB’s symbolic toolbox.

4) \textbf{prodint} \((l = 0, u = 1)\)

Assuming \( u = 1 \) and \( l = 0 \) so that \( x, \eta \in [0,1] \) and letting \( l^* = \max(0, \eta - \theta) \) and
\( u^* = \min(1, \eta + \theta) \), we have

\[
\text{prodint}(0, 1, \eta; k, \theta) = \int_0^1 x^k R_Y(x, \eta; \theta) \, dx
\]

\[
= \int_{u^*}^\eta x^k \left\{ \left( 1 - \frac{\eta - x}{\theta} \right) \cos \left( \frac{\pi(\eta - x)}{\theta} \right) + \frac{1}{\pi} \sin \left( \frac{\pi(\eta - x)}{\theta} \right) \right\} \, dx
\]

\[
+ \int_\eta^{u^*} x^k \left\{ \left( 1 - \frac{x - \eta}{\theta} \right) \cos \left( \frac{\pi(x - \eta)}{\theta} \right) + \frac{1}{\pi} \sin \left( \frac{\pi(x - \eta)}{\theta} \right) \right\} \, dx
\]

\[
= \text{ptint}(\eta, \theta, -1, k, l^*, \eta) + \text{ptint}(\eta, \theta, -1, k, \eta, u^*)
\]

where

\[
\text{ptint}(d, t, s, k, a, b) \equiv \int_a^b x^k \left\{ \left( 1 - \frac{d + sx}{t} \right) \cos \left( \frac{\pi(d + sx)}{t} \right) + \frac{1}{\pi} \sin \left( \frac{\pi(d + sx)}{t} \right) \right\} \, dx
\]

for \( d > 0, t > 0, s \in \{-1, 1\}, k \in \{0, 1, 2, \ldots\} \) and \( b > a > 0 \). To simplify \( \text{ptint} \), we first use the basic angle sum and difference identities \((4.3.9) - (4.3.10)\) to obtain

\[
\text{ptint}(d, t, s, k, a, b)
\]

\[
= \int_a^{b'} \left( 1 - \frac{d}{t} \right) \cos \left( \frac{d t}{\pi} \right) x^k \cos \left( \frac{sx}{t} \pi \right) \, dx
\]

\[
- \int_a^{b'} \left( 1 - \frac{d}{t} \right) \sin \left( \frac{d t}{\pi} \right) x^k \sin \left( \frac{sx}{t} \pi \right) \, dx
\]

\[
- \int_a^{b'} \frac{s}{t} \cos \left( \frac{d t}{\pi} \right) x^{k+1} \cos \left( \frac{sx}{t} \pi \right) \, dx + \int_a^{b'} \frac{s}{t} \sin \left( \frac{d t}{\pi} \right) x^{k+1} \sin \left( \frac{sx}{t} \pi \right) \, dx
\]

\[
+ \int_a^{b'} \frac{1}{\pi} \sin \left( \frac{d t}{\pi} \right) x^k \cos \left( \frac{sx}{t} \pi \right) \, dx + \int_a^{b'} \frac{1}{\pi} \cos \left( \frac{d t}{\pi} \right) x^k \sin \left( \frac{sx}{t} \pi \right) \, dx.
\]

Making the change of variable \( v = \frac{sx}{t} \pi \) and letting \( a' = \frac{sa}{t} \pi \) and \( b' = \frac{sb}{t} \pi \), we have

\[
\text{ptint}(d, t, s, k, a, b) = \int_{a'}^{b'} \left( 1 - \frac{d}{t} \right) \cos \left( \frac{d t}{\pi} \right) \left( \frac{t}{\pi s} \right)^{k+1} v^k \cos(v) \, dv
\]

\[
- \int_{a'}^{b'} \left( 1 - \frac{d}{t} \right) \sin \left( \frac{d t}{\pi} \right) \left( \frac{t}{\pi s} \right)^{k+1} v^k \sin(v) \, dv
\]

\[
- \int_{a'}^{b'} \frac{s}{t} \cos \left( \frac{d t}{\pi} \right) \left( \frac{t}{\pi s} \right)^{k+2} v^{k+1} \cos(v) \, dv
\]

\[
+ \int_{a'}^{b'} \frac{1}{\pi} \sin \left( \frac{d t}{\pi} \right) \left( \frac{t}{\pi s} \right)^{k+1} v^{k+1} \sin(v) \, dv.
\]
\[ + \int_{a'}^{b'} \frac{s}{t} \sin \left( \frac{d}{\pi} \right) \left( \frac{t}{\pi s} \right)^{k+2} v^{k+1} \sin (v) \, dv \]
\[ + \int_{a'}^{b'} \frac{1}{\pi} \sin \left( \frac{d}{\pi} \right) \left( \frac{t}{\pi s} \right)^{k+1} v^k \cos (v) \, dv \]
\[ + \int_{a'}^{b'} \frac{1}{\pi} \cos \left( \frac{d}{\pi} \right) \left( \frac{t}{\pi s} \right)^{k+1} v^k \sin (v) \, dv. \]

Using the recursions
\[ \int v^k \sin (v) \, dv = -v^k \cos (v) + k \int v^{n-1} \cos (v) \, dv \]
\[ \int v^k \cos (v) \, dv = v^k \sin (v) - k \int v^{n-1} \sin (v) \, dv, \]
we can express \( ptint \) as
\[
ptint (d, t, s, k, a, b) = \left( 1 - \frac{d}{t} \right) \cos \left( \frac{d}{\pi} \right) \left( \frac{t}{\pi s} \right)^{k+1} \cosprodint (k, a', b') \]
\[
- \left( 1 - \frac{d}{t} \right) \sin \left( \frac{d}{\pi} \right) \left( \frac{t}{\pi s} \right)^{k+1} \sinprodint (k, a', b') \]
\[- \frac{s}{t} \cos \left( \frac{d}{\pi} \right) \left( \frac{t}{\pi s} \right)^{k+2} \cosprodint (k+1, a', b') \]
\[+ \frac{s}{t} \sin \left( \frac{d}{\pi} \right) \left( \frac{t}{\pi s} \right)^{k+2} \sinprodint (k+1, a', b') \]
\[+ \frac{1}{\pi} \sin \left( \frac{d}{\pi} \right) \left( \frac{t}{\pi s} \right)^{k+1} \cosprodint (k, a', b') \]
\[+ \frac{1}{\pi} \cos \left( \frac{d}{\pi} \right) \left( \frac{t}{\pi s} \right)^{k+1} \sinprodint (k, a', b'), \]

where \( \cosprodint \) and \( \sinprodint \) are defined recursively as

\[
\cosprodint (k, a', b') \]
\[= \left\{ \begin{array}{ll}
(b')^k \sin (b') - (a')^k \sin (a') - k \sinprodint (k - 1, a', b'), & k \geq 1; \\
\sin (b') - \sin (a'), & k = 0
\end{array} \right. \]

and

\[
\sinprodint (k, a', b') \]
\[= \left\{ \begin{array}{ll}
(a')^k \cos (a') - (b')^k \cos (b') + k \cosprodint (k - 1, a', b'), & k \geq 1; \\
\cos (a') - \cos (b'), & k = 0.
\end{array} \right. \]

263
4.3.3 Cubic Correlation Function

The cubic correlation function is

\[
R_Y(x, \eta; \theta) = \begin{cases} 
1 - 6 \left( \frac{x-\eta}{\theta} \right)^2 + 6 \left( \frac{|x-\eta|}{\theta} \right)^3, & |x-\eta| \leq \frac{\theta}{2}, \\
2 \left( 1 - \frac{|x-\eta|}{\theta} \right)^3, & \frac{\theta}{2} \leq |x-\eta| \leq \theta; \\
0, & \theta < |x-\eta|. 
\end{cases} \tag{4.3.11}
\]

Moon (2010) derives closed form expressions for \text{sgint}, \text{dbint}, and \text{mxint}, and we will overview the simplification of these three integrals. Then, in order to estimate sensitivity indices with an arbitrary polynomial mean, we will show how to derive a closed form expression for \text{prodint}.

1) \text{sgint}

Let

\[
R_1(x, \eta) \equiv R_Y^1(x, \eta; \theta) = 1 - 6 \left( \frac{x-\eta}{\theta} \right)^2 + 6 \left( \frac{|x-\eta|}{\theta} \right)^3
\]

\[
R_2(x, \eta) \equiv R_Y^2(x, \eta; \theta) = 2 \left( 1 - \frac{|x-\eta|}{\theta} \right)^3
\tag{4.3.12}
\]

denote the possible forms of cubic correlation function. Also, let \( l^* = \max(\eta - \theta, l) \), \( u^* = \min(\eta + \theta, u) \), and \( A \equiv \int_{l^*}^{u} R_Y(x, \eta; \theta) dx \) so that \text{sgint}(l, u, \eta; \theta) = \frac{A}{u-l}. \) Then, \( A \) is solved as follows.

Case 1 : \( l > \eta - \frac{\theta}{2} \) and \( u \leq \eta + \frac{\theta}{2} \)

\[
A = \int_{l^*}^{u} R_1(x, \eta) dx = (u-l) - \frac{2(u-\eta)^3 - (u-\eta)^4}{6\theta^2} + \frac{3(u-\eta)^4 + (u-\eta)^4}{2\theta^3}
\]

Case 2 : \( l > \eta - \frac{\theta}{2} \) and \( u > \eta + \frac{\theta}{2} \)

\[
A = \int_{l}^{\eta + \frac{\theta}{2}} R_1(x, \eta) dx + \int_{\eta + \frac{\theta}{2}}^{u} R_2(x, \eta) dx = (\eta-l) + \frac{3\theta}{8} + \frac{2(l-\eta)^3 + 3(l-\eta)^4 - (\theta-u^*+\eta)^4}{2\theta^3}
\]

Case 3 : \( l \leq \eta - \frac{\theta}{2} \) and \( u \leq \eta + \frac{\theta}{2} \)

\[
A = \int_{l}^{\eta - \frac{\theta}{2}} R_2(x, \eta) dx + \int_{\eta - \frac{\theta}{2}}^{u} R_1(x, \eta) dx = (u-\eta) + \frac{3\theta}{8} - \frac{2(u-\eta)^3 + 3(u-\eta)^4 - (\theta+u^*-\eta)^4}{2\theta^3}
\]
As in (4.3.12), denote possible forms of the cubic correlation function by

\[ A = \int_{l-t}^{\eta - \frac{\theta}{2}} R_2(x, \eta)dx + \int_{\eta - \frac{\theta}{2}}^{\eta + \frac{\theta}{2}} R_1(x, \eta)dx + \int_{\eta + \frac{\theta}{2}}^{u+0} R_2(x, \eta)dx = \frac{3\theta}{4} \left( \frac{\theta + u - \eta}{2} \right)^4 \]

2) dbint

Let \( B \equiv \int_{l}^{u} \int_{u}^{R^Y(x, \eta; \theta)d\eta dx, \) then \( \text{dbint}(l, u; \theta) = \frac{B}{(u-l)^\theta}. \) B is solved in the following 3 cases.

Case 1 : \( u - l > \theta \)

\[ B = \int_{l}^{u-\theta} \int_{x+\frac{\theta}{2}}^{x+\theta} R_2(x, \eta)d\eta dx + \int_{u-\theta}^{u} \int_{x+\frac{\theta}{2}}^{x+\theta} R_2(x, \eta)d\eta dx + \int_{u}^{u+\theta} \int_{x+\frac{\theta}{2}}^{x+\theta} R_2(x, \eta)d\eta dx + \int_{l+\theta}^{u+\theta} \int_{x-\frac{\theta}{2}}^{x+\theta} R_2(x, \eta)d\eta dx + \int_{l}^{u+\theta} \int_{x-\frac{\theta}{2}}^{x-\theta} R_1(x, \eta)d\eta dx + \int_{l+\theta}^{u} \int_{x-\frac{\theta}{2}}^{x-\theta} R_1(x, \eta)d\eta dx + \int_{l}^{u} \int_{x-\frac{\theta}{2}}^{x-\theta} R(x, \eta)d\eta dx \]

\[ = \frac{23}{40} \theta^2 + \frac{3}{4} \theta(-l - \theta + u) \]

Case 2 : \( \frac{\theta}{2} < u - l \leq \theta \)

\[ B = \int_{l}^{u-\frac{\theta}{2}} \int_{x+\frac{\theta}{2}}^{x+\theta} R_2(x, \eta)d\eta dx + \int_{u-\frac{\theta}{2}}^{u} \int_{x+\frac{\theta}{2}}^{x+\theta} R_2(x, \eta)d\eta dx + \int_{u}^{u+\frac{\theta}{2}} \int_{x+\frac{\theta}{2}}^{x+\theta} R_2(x, \eta)d\eta dx + \int_{l+\frac{\theta}{2}}^{u+\frac{\theta}{2}} \int_{x-\frac{\theta}{2}}^{x+\theta} R_1(x, \eta)d\eta dx + \int_{l}^{u+\frac{\theta}{2}} \int_{x-\frac{\theta}{2}}^{x-\theta} R_1(x, \eta)d\eta dx + \int_{l+\frac{\theta}{2}}^{u} \int_{x-\frac{\theta}{2}}^{x-\theta} R_1(x, \eta)d\eta dx \]

\[ = \frac{(l+\theta-u)^5}{5\theta^3} + \frac{3}{4} \theta(u-l) - \frac{7}{40} \theta^2 \]

Case 3 : \( u - l \leq \frac{\theta}{2} \)

\[ B = \int_{l}^{u} \int_{l}^{u} R_1(x, \eta)d\eta dx = (l-u)^2 - \frac{(l-u)(3l-3u+5\theta)}{5\theta^3} \]

3) mxint

As in (4.3.12), denote possible forms of the cubic correlation function by

\[ R_1(\eta_1) \equiv R_1^Y(x, \eta_1; \theta) = 1 - 6 \left( \frac{x-\eta_1}{\theta} \right)^2 + 6 \left( \frac{|x-\eta_1|}{\theta} \right)^3 \]

\[ R_2(\eta_1) \equiv R_2^Y(x, \eta_1; \theta) = 2 \left( 1 - \frac{|x-\eta_1|}{\theta} \right)^3 \]

\[ R_1(\eta_2) \equiv R_1^Y(x, \eta_2; \theta) = 1 - 6 \left( \frac{x-\eta_2}{\theta} \right)^2 + 6 \left( \frac{|x-\eta_2|}{\theta} \right)^3 \]

\[ R_2(\eta_2) \equiv R_2^Y(x, \eta_2; \theta) = 2 \left( 1 - \frac{|x-\eta_2|}{\theta} \right)^3 \]

265
Let \( C \equiv \int_l^\infty R^x(x, \eta_1; \theta)R^y(x, \eta_2; \theta)dx \), then \( \text{mxint}(l, u, \eta_1, \eta_2; \theta) = \frac{C}{u-l} \). Assume \( \eta_1 \leq \eta_2 \) and let \( \delta = \eta_2 - \eta_1 \), \( l^* = \max(\eta_2 - \theta, l) \), and \( u^* = \min(\eta_1 + \theta, u) \). The simplification of the \( C \) must be broken down into several different regions of integration, since the integral has a different form depending on the location of \( l, u, \) and the magnitude of \( \delta \). We will present the form of the integral for each case. In all cases, these integrals are polynomials and can be calculated in a straightforward manner.

Case 1 : \( \delta = 0 \)

(a) if \( l > \eta_1 - \frac{\theta}{2} \) and \( u \leq \eta_1 + \frac{\theta}{2} \),

\[
C = \int_l^u R_1(\eta_1)R_1(\eta_2)dx
\]

(b) if \( l > \eta_1 - \frac{\theta}{2} \) and \( u > \eta_1 + \frac{\theta}{2} \),

\[
C = \int_{\eta_1 + \frac{\theta}{2}}^{\eta_1 + \frac{\theta}{2}} R_1(\eta_1)R_1(\eta_2)dx + \int_{\eta_1 + \frac{\theta}{2}}^{\eta_1 + \frac{\theta}{2}} R_2(\eta_1)R_2(\eta_2)dx
\]

(c) if \( l \leq \eta_1 - \frac{\theta}{2} \) and \( u \leq \eta_1 + \frac{\theta}{2} \),

\[
C = \int_{\eta_1 - \frac{\theta}{2}}^{\eta_1 - \frac{\theta}{2}} R_2(\eta_1)R_2(\eta_2)dx + \int_{\eta_1 - \frac{\theta}{2}}^{\eta_1 - \frac{\theta}{2}} R_1(\eta_1)R_1(\eta_2)dx
\]

(d) if \( l \leq \eta_1 - \frac{\theta}{2} \) and \( u > \eta_1 + \frac{\theta}{2} \),

\[
C = \int_{\eta_1 - \frac{\theta}{2}}^{\eta_1 - \frac{\theta}{2}} R_2(\eta_1)R_2(\eta_2)dx + \int_{\eta_1 - \frac{\theta}{2}}^{\eta_1 - \frac{\theta}{2}} R_1(\eta_1)R_1(\eta_2)dx + \int_{\eta_1 - \frac{\theta}{2}}^{\eta_1 - \frac{\theta}{2}} R_2(\eta_1)R_2(\eta_2)dx
\]

Case 2 : \( 0 < \delta \leq \frac{\theta}{2} \)

(a) if \( l > \eta_2 - \frac{\theta}{2} \) and \( u \leq \eta_1 + \frac{\theta}{2} \),

\[
C = \int_l^u R_1(\eta_1)R_1(\eta_2)dx
\]

(b) if \( l > \eta_2 - \frac{\theta}{2} \) and \( \eta_1 + \frac{\theta}{2} < u \leq \eta_2 + \frac{\theta}{2} \),

\[
C = \int_{\eta_2 + \frac{\theta}{2}}^{\eta_2 + \frac{\theta}{2}} R_1(\eta_1)R_1(\eta_2)dx + \int_{\eta_2 + \frac{\theta}{2}}^{\eta_2 + \frac{\theta}{2}} R_2(\eta_1)R_2(\eta_2)dx
\]

(c) if \( l > \eta_2 - \frac{\theta}{2} \) and \( u > \eta_2 + \frac{\theta}{2} \),

\[
C = \int_{\eta_2 + \frac{\theta}{2}}^{\eta_2 + \frac{\theta}{2}} R_1(\eta_1)R_1(\eta_2)dx + \int_{\eta_2 + \frac{\theta}{2}}^{\eta_2 + \frac{\theta}{2}} R_2(\eta_1)R_2(\eta_2)dx + \int_{\eta_2 + \frac{\theta}{2}}^{\eta_2 + \frac{\theta}{2}} R_2(\eta_1)R_2(\eta_2)dx
\]
(d) if $\eta_1 - \frac{\theta}{2} < l \leq \eta_2 - \frac{\theta}{2}$ and $u \leq \eta_1 + \frac{\theta}{2}$,

$$C = \int_{l}^{\eta_2 - \frac{\theta}{2}} R_1(\eta_1) R_2(\eta_2)\,dx + \int_{\eta_2 - \frac{\theta}{2}}^{u} R_1(\eta_1) R_1(\eta_2)\,dx$$

(e) if $\eta_1 - \frac{\theta}{2} < l \leq \eta_2 - \frac{\theta}{2}$ and $\eta_1 + \frac{\theta}{2} < u \leq \eta_2 + \frac{\theta}{2}$,

$$C = \int_{l}^{\eta_2 - \frac{\theta}{2}} R_1(\eta_1) R_2(\eta_2)\,dx + \int_{\eta_2 - \frac{\theta}{2}}^{\eta_1 + \frac{\theta}{2}} R_1(\eta_1) R_1(\eta_2)\,dx + \int_{\eta_1 + \frac{\theta}{2}}^{u} R_2(\eta_1) R_1(\eta_2)\,dx$$

(f) if $\eta_1 - \frac{\theta}{2} < l \leq \eta_2 - \frac{\theta}{2}$ and $u > \eta_2 + \frac{\theta}{2}$,

$$C = \int_{l}^{\eta_2 - \frac{\theta}{2}} R_1(\eta_1) R_2(\eta_2)\,dx + \int_{\eta_2 - \frac{\theta}{2}}^{\eta_1 + \frac{\theta}{2}} R_1(\eta_1) R_1(\eta_2)\,dx + \int_{\eta_1 + \frac{\theta}{2}}^{u} R_2(\eta_1) R_1(\eta_2)\,dx + \int_{u}^{\eta_2 + \frac{\theta}{2}} R_2(\eta_1) R_2(\eta_2)\,dx$$

(g) if $l < \eta_1 - \frac{\theta}{2}$ and $u \leq \eta_1 + \frac{\theta}{2}$,

$$C = \int_{l}^{\eta_1 - \frac{\theta}{2}} R_2(\eta_1) R_2(\eta_2)\,dx + \int_{\eta_1 - \frac{\theta}{2}}^{\eta_2 - \frac{\theta}{2}} R_1(\eta_1) R_2(\eta_2)\,dx + \int_{\eta_2 - \frac{\theta}{2}}^{u} R_1(\eta_1) R_1(\eta_2)\,dx$$

(h) if $l < \eta_1 - \frac{\theta}{2}$ and $\eta_1 + \frac{\theta}{2} < u \leq \eta_2 + \frac{\theta}{2}$,

$$C = \int_{l}^{\eta_1 - \frac{\theta}{2}} R_2(\eta_1) R_2(\eta_2)\,dx + \int_{\eta_1 - \frac{\theta}{2}}^{\eta_2 - \frac{\theta}{2}} R_1(\eta_1) R_2(\eta_2)\,dx + \int_{\eta_2 - \frac{\theta}{2}}^{\eta_1 + \frac{\theta}{2}} R_1(\eta_1) R_1(\eta_2)\,dx + \int_{\eta_1 + \frac{\theta}{2}}^{u} R_2(\eta_1) R_1(\eta_2)\,dx$$

(i) if $l < \eta_1 - \frac{\theta}{2}$ and $u > \eta_2 + \frac{\theta}{2}$,

$$C = \int_{l}^{\eta_1 - \frac{\theta}{2}} R_2(\eta_1) R_2(\eta_2)\,dx + \int_{\eta_1 - \frac{\theta}{2}}^{\eta_2 - \frac{\theta}{2}} R_1(\eta_1) R_2(\eta_2)\,dx + \int_{\eta_2 - \frac{\theta}{2}}^{\eta_1 + \frac{\theta}{2}} R_1(\eta_1) R_1(\eta_2)\,dx + \int_{\eta_1 + \frac{\theta}{2}}^{u} R_2(\eta_1) R_2(\eta_2)\,dx + \int_{u}^{\eta_2 + \frac{\theta}{2}} R_2(\eta_1) R_2(\eta_2)\,dx$$

Case 3: $\frac{\theta}{2} < \delta \leq \theta$

$$C = \int_{l}^{\eta_2 - \frac{\theta}{2}} R_1(\eta_1) R_2(\eta_2)\,dx + \int_{\eta_2 - \frac{\theta}{2}}^{\eta_1 + \frac{\theta}{2}} R_1(\eta_1) R_1(\eta_2)\,dx + \int_{\eta_1 + \frac{\theta}{2}}^{u} R_2(\eta_1) R_1(\eta_2)\,dx$$

Case 4: $\theta < \delta \leq \frac{3}{2}\theta$

$$C = \int_{\eta_2 - \frac{\theta}{2}}^{\eta_1 + \frac{\theta}{2}} R_1(\eta_1) R_2(\eta_2)\,dx + \int_{\eta_1 + \frac{\theta}{2}}^{\eta_2 - \frac{\theta}{2}} R_2(\eta_1) R_2(\eta_2)\,dx + \int_{\eta_2 - \frac{\theta}{2}}^{\eta_1 + \frac{\theta}{2}} R_2(\eta_1) R_1(\eta_2)\,dx + \int_{\eta_1 + \frac{\theta}{2}}^{u} R_1(\eta_1) R_1(\eta_2)\,dx$$

Case 5: $\frac{3}{2}\theta < \delta \leq 2\theta$

$$C = \int_{\eta_2 - \frac{\theta}{2}}^{\eta_2 + \frac{\theta}{2}} R_2(\eta_1) R_2(\eta_2)\,dx$$
4) prodint \((l = 0, u = 1)\)

Just as in sgint, the integral of \(R^Y(x, \eta; \theta)\) with respect to \(x\) depends on the value of \(\eta\) in determining the form of correlation function. Assuming \(l = 0\) and \(u = 0\), let \(l^* = \max(\eta - \theta, 0)\), \(u^* = \min(\eta + \theta, 1)\), and \(R_1(x, \eta)\) and \(R_2(x, \eta)\) be defined in (4.3.12). prodint is then solved as follows.

**Case 1:** \(0 > \eta - \frac{\theta}{2}\) and \(1 \leq \eta + \frac{\theta}{2}\)

\[
\text{prodint}(0, 1, \eta; k, \theta) = \int_0^1 x^k R_1(x, \eta)\,dx
\]

\[
= \int_0^\eta x^k \left\{1 - 6 \left(\frac{x - \eta}{\theta}\right)^2 + 6 \left(\frac{\eta - x}{\theta}\right)^3\right\} \,dx + \int_\eta^1 x^k \left\{1 - 6 \left(\frac{x - \eta}{\theta}\right)^2 + 6 \left(\frac{x - \eta}{\theta}\right)^3\right\} \,dx
\]

**Case 2:** \(0 > \eta - \frac{\theta}{2}\) and \(1 > \eta + \frac{\theta}{2}\)

\[
\text{prodint}(0, 1, \eta; k, \theta) = \int_0^{\eta + \frac{\theta}{2}} x^k R_1(x, \eta)\,dx + \int_{\eta + \frac{\theta}{2}}^{u^*} x^k R_2(x, \eta)\,dx
\]

\[
= \int_0^\eta x^k \left\{1 - 6 \left(\frac{x - \eta}{\theta}\right)^2 + 6 \left(\frac{\eta - x}{\theta}\right)^3\right\} \,dx + \int_\eta^{\eta + \frac{\theta}{2}} x^k \left\{1 - 6 \left(\frac{x - \eta}{\theta}\right)^2 + 6 \left(\frac{x - \eta}{\theta}\right)^3\right\} \,dx + \int_{\eta + \frac{\theta}{2}}^{u^*} x^k \left\{2 \left(1 - \frac{x - \eta}{\theta}\right)^3\right\} \,dx
\]

**Case 3:** \(0 \leq \eta - \frac{\theta}{2}\) and \(1 \leq \eta + \frac{\theta}{2}\)

\[
\text{prodint}(0, 1, \eta; k, \theta) = \int_{l^*}^{\eta - \frac{\theta}{2}} x^k R_2(x, \eta)\,dx + \int_{\eta - \frac{\theta}{2}}^1 x^k R_1(x, \eta)\,dx
\]

\[
= \int_{l^*}^{\eta - \frac{\theta}{2}} x^k \left\{2 \left(1 - \frac{\eta - x}{\theta}\right)^3\right\} \,dx + \int_{\eta - \frac{\theta}{2}}^\eta x^k \left\{1 - 6 \left(\frac{x - \eta}{\theta}\right)^2 + 6 \left(\frac{\eta - x}{\theta}\right)^3\right\} \,dx + \int_{\eta - \frac{\theta}{2}}^{1} x^k \left\{1 - 6 \left(\frac{x - \eta}{\theta}\right)^2 + 6 \left(\frac{\eta - x}{\theta}\right)^3\right\} \,dx
\]
\[ + \int_{\eta}^{1} x^k \left\{ 1 - 6 \left( \frac{x - \eta}{\theta} \right)^2 + 6 \left( \frac{x - \eta}{\theta} \right)^3 \right\} \, dx \]

Case 4: \(0 \leq \eta - \frac{\theta}{2}\) and \(1 > \eta + \frac{\theta}{2}\)

\[ \text{prodint}(0, 1, \eta; k, \theta) = \int_{l^*}^{\eta - \frac{\theta}{2}} x^k R_2(x, \eta) \, dx + \int_{\eta - \frac{\theta}{2}}^{\eta + \frac{\theta}{2}} x^k R_1(x, \eta) \, dx + \int_{\eta + \frac{\theta}{2}}^{u^*} x^k R_2(x, \eta) \, dx \]

Notice that in all cases, \text{prodint} is just the sum of integrals of polynomials. The simplification of these integrals, while tedious, can be done in a straightforward manner. Our implementation will rely on analytic expressions found via MATLAB’s symbolic toolbox.

4.4 Examples
4.4.1 A Simple Two Dimensional Function

As a simple example, consider the function

\[ y(x) = (x_1 - 0.5)^2 + (x_2 - 0.5)^2 + (x_1 - 0.5)(x_2 - 0.5) + 0.05 \sin(4\pi x_1) + 0.05 \sin(4\pi x_2), \quad (4.4.1) \]

which was first introduced in Section 4.1.3. Here, we will analytically calculate the sensitivity indices of \(y(x)\), and then compare them to the estimated sensitivity indices.
based on function evaluations at the ten-point Latin hypercube design from Figure 4.2. We will estimate the sensitivity indices using both a GP with a constant mean and the quadratic mean specified by (4.1.6), and we will employ all three correlation function (Gaussian, Bohman, and cubic) discussed in Section 4.3. The goal here is to show that modeling a known trend in a computer experiment can lead to improved prediction and better estimates of sensitivity indices when the initial design is small.

**Analytical Calculation of Main Effect and Total Effect Sensitivity Indices**

Notice that

\[
\begin{align*}
y_0 &= \int_{[0,1]^2} y(\mathbf{x}) d\mathbf{x} = \frac{1}{6} \\
y_1(x_1) &= \int_0^1 y(\mathbf{x}) dx_1 - y_0 = \frac{\sin(4\pi x - 1)}{20} + x_1^2 - x_1 + \frac{1}{6} \\
y_1(x_1) &= \int_0^1 y(\mathbf{x}) dx_2 - y_0 = \frac{\sin(4\pi x - 2)}{20} + x_2^2 - x_2 + \frac{1}{6}.
\end{align*}
\]

So, the joint effect variances can be shown to be

\[
\begin{align*}
V &= \int_{[0,1]^2} y^2(\mathbf{x}) d\mathbf{x} - y_0^2 = \frac{29}{600} - \frac{1}{36} = \frac{37}{1800} \\
V_1 &= \int_0^1 y_1^2(\mathbf{x}) dx_1 = \frac{49}{7200} \\
V_2 &= \int_0^1 y_2^2(\mathbf{x}) dx_2 = \frac{49}{7200} \\
V_{12} &= V - V_1 - V_2 = \frac{1}{144}.
\end{align*}
\]

Therefore, the main effect and total effect sensitivity indices are

\[
\begin{align*}
S_1 &= \frac{V_1}{V} = \frac{49}{148} \approx 0.3311 \\
S_2 &= \frac{V_2}{V} = \frac{49}{148} \approx 0.3311 \\
T_1 &= \frac{V_1 + V_{12}}{V} = \frac{99}{148} \approx 0.6689 \\
T_2 &= \frac{V_2 + V_{12}}{V} = \frac{99}{148} \approx 0.6689.
\end{align*}
\]
Table 4.1: Estimated main and total effect sensitivity indices for (4.4.1) using all three correlation functions and both constant and quadratic mean structures, as well as the ERMSPEs for the predictor of (4.4.1) produced by each GP model.

We will use these analytically derived indices as a basis of comparison for the indices estimated using both a constant and quadratic mean GP with three different correlation functions.

**Estimation of Main Effect and Total Effect Sensitivity Indices**

Using the approach described in Section 4.2, we estimate the sensitivity indices after evaluating $y(\cdot)$ at our initial ten-point Latin hypercube design, using all three correlation function and both constant and quadratic means. The estimated sensitivity indices, as well as the ERMSPEs based on a dense $50 \times 50$ grid of validation points in $[0,1]^2$ are summarized in Table 4.4.1.

For all correlation functions, we obtain similar results. Both mean functions (constant and quadratic) seem to underestimate the main effects and overestimate the total effects; this means that the joint effect sensitivity index for the interaction of the first and second input is overestimated. However, the magnitude of overestimation and underestimation is much smaller when using the quadratic mean (4.1.6), as the estimated indices are closer to the analytically derived indices. Additionally, the
Table 4.2: Estimated sensitivity indices for the borehole function based on a 160-point space-filling Latin hypercube design, a constant mean GP model, and an unrestricted correlation parameter space.

ERMSPEs are always smaller when using the quadratic mean than when using the constant. This seems to suggest that, for this particular example which has a strong quadratic trend, it is advantageous to using a GP with a quadratic mean for both prediction and estimation of sensitivity indices.

4.4.2 Borehole Function

Now, we will focus on the borehole function, which was presented in Section 4.1.3. We will assume that we have a large Latin hypercube design of 400 inputs. We will fit a GP model to the functions using both the standard constant mean, Gaussian correlation assumptions and the recently-introduced approach of Kaufman et al. (2010) that relies on a higher-degree polynomial mean, a compactly supported correlation function with a restricted parameter space, and computationally efficient spare matrix methods. The goal is to show that we can obtain good estimates of the main and total effect sensitivity indices using the the latter approach while substantially reducing the computational burden.

First, we will create a baseline for judging the accuracy of the estimates of the sensitivity indices. We created a space-filling 160 point Latin hypercube design in
the eight-dimensional input space (20 observations per dimension), and fit a constant mean GP model using function evaluations at this initial design. (This design is small enough so that the covariance matrix is not near singular and is invertible, while also being large enough to accurately emulate the borehole function.) Then, the sensitivity indices were estimated using the initial design, function evaluations, and fitted GP model. The results are presented in Table 4.2. Judging by these computations, we can see that \( x_1 \) (the radius of the borehole) is the most influential input on the flow rate of the borehole. Also, \( x_5, x_6, x_7, \) and \( x_8 \) (the transmissivity of the lower aquifer, the potentiometric head of the lower aquifer, the length of the borehole, and the hydraulic conductivity, respectively) all have a small influence on the output, while the \( x_1, x_3, \) and \( x_4 \) (the radius of influence, the transmissivity of the upper aquifer, and the potentiometric head of the upper aquifer) all appear to have a negligible effect on the output. Additionally, in absolute terms, the main effects seem to account for most of the variation in the flow rate, but for some of the inputs with smaller effects, these interactions appear to be somewhat substantial relative to the magnitude of the main effects. The estimations were made using all three (Gaussian, Bohman, and cubic) correlation functions, and all three choices yielded similar results, confirming that these baseline values are independent of the fitted correlation function.

With this baseline for the true sensitivity indices, we can compare the different estimation approaches with the 400-point dataset. The first approach is to use the standard constant mean GP with noted earlier, JMP deals with the almost-singular covariance matrix by adding a small amount of measurement error, which makes the predictor of \( y(\mathbf{x}) \) a smoother of the original data instead of an interpolator. Additionally, JMP utilizes a plug-in approach to estimate the joint effect variances
Table 4.3: Details of the various implementations for estimating sensitivity indices with the 400-point design. α is the maximum possible proportion of off-diagonal elements of $\Sigma_{z_{sim}}$ that can be non-zero when restricting the parameters space of the correlation parameters as described in (4.1.7)-(4.1.9) (applicable for compactly supported correlation function only), ERMSPE is calculated using a set of 200 validations points, and Time measures how long it took to fit the model and obtain estimates of sensitivity indices on an Intel Core 2 Duo CPU 3.00 with 4 GB RAM. The estimates of the sensitivity indices can be found in Table 4.4.

$$(\hat{V}_j^j = Var_g [E_g \{\hat{g}(X)|X_S]\} )$$. The second and third approach both use a polynomial mean that includes a fifth degree polynomial of each input plus all first-order interactions, a compactly supported correlation function, and a restricted parameter space on the correlation parameters to ensure sparsity. (Specifically, at most 10% of the off diagonal elements of $\Sigma_{z_{sim}}$ can be non-zero.) Both are implemented in the MATLAB program MPErK (see Han et al. (2011)), which utilizes the plug-in Bayesian approach to estimating joint effect variances outlined in Section 4.2 (i.e., $\hat{V}_j^j = E_p \{Var_g [E_g Y(X)|X_S]|Z_{sim}\}$), and both fit a model with no measurement error. The only difference is the choice of compactly supported correlation function; ones uses the Bohman, while the other uses the cubic. All three approaches estimate the unknown Gaussian process parameters via maximum likelihood.

Judging by the estimated sensitivity indices in Table 4.4, it appears that all three approaches produce reasonable results, as none of these estimated indices deviate too
Table 4.4: Estimated sensitivity indices for the Borehole function for an initial 400-point space-filling Latin hypercube design. The details of the various correlation function and software choices can be found in Table 4.3.

greatly from the baseline indices in Table 4.2. Therefore, to separate the three approaches, it is more useful to look at some other factors presented in Table 4.3. There appears to be a slight advantage in predictive ability in the JMP’s Gaussian implementation, as the ERMSPE at 200 validation points is just over 1.1, while the implementations with compactly supported correlation functions both have ERMSPEs over 1.4. Considering the range of $y(\cdot)$, this difference may be of little practical significance. There are two main advantages to the sparse covariance, polynomial mean models with Bohman and cubic correlation functions. First, both methods interpolate the training data, which is not true of JMP’s Gaussian implementation. Second, the time required to fit the model and estimate the sensitivity indices on an Intel Core 2 Duo CPU 3.00 with 4 GB RAM is almost three times as long when using the constant mean Gaussian correlated GP. This is probably due to the fact that the evaluation of the likelihood function with a Gaussian correlation always requires inversion of a

<table>
<thead>
<tr>
<th>Corr./Software</th>
<th>Effect</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$x_6$</th>
<th>$x_7$</th>
<th>$x_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian/JMP</td>
<td>Main</td>
<td>0.0000</td>
<td>0.8329</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0408</td>
<td>0.0411</td>
<td>0.0389</td>
<td>0.0117</td>
</tr>
<tr>
<td>Gaussian/JMP</td>
<td>Total</td>
<td>0.0000</td>
<td>0.8685</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0526</td>
<td>0.0530</td>
<td>0.0505</td>
<td>0.0119</td>
</tr>
<tr>
<td>Bohman/MPErK</td>
<td>Main</td>
<td>0.0000</td>
<td>0.8326</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0405</td>
<td>0.0408</td>
<td>0.0397</td>
<td>0.0085</td>
</tr>
<tr>
<td>Bohman/MPErK</td>
<td>Total</td>
<td>0.0006</td>
<td>0.8688</td>
<td>0.0007</td>
<td>0.0004</td>
<td>0.0529</td>
<td>0.0534</td>
<td>0.0520</td>
<td>0.0123</td>
</tr>
<tr>
<td>Cubic/MPErK</td>
<td>Main</td>
<td>0.0000</td>
<td>0.8326</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0405</td>
<td>0.0408</td>
<td>0.0397</td>
<td>0.0085</td>
</tr>
<tr>
<td>Cubic/MPErK</td>
<td>Total</td>
<td>0.0005</td>
<td>0.8689</td>
<td>0.0006</td>
<td>0.0006</td>
<td>0.0528</td>
<td>0.0534</td>
<td>0.0519</td>
<td>0.0123</td>
</tr>
</tbody>
</table>
400 × 400 matrix with no zeros off the diagonal, while the methods specifically made
to handle large designs only require inversion of a 400 × 400 sparse matrix, and the
inversion of the $69 \times 69 \text{ } F^\top \left( \Sigma_{sim}^Z \right)^{-1} F$ matrix. Note that it is not so straightforward
to compare the run-times of the Gaussian/JMP estimates with the Bohman/MPErK
and cubic/MPErK estimates, as the former is implemented in C while the latter is
implemented in MATLAB, a slower but more user-friendly language. Knowing this,
the quicker run-times in MPErK using sparse matrix methods look even more impres-
sive when compared to the run-times from JMP, which do not utilize sparse matrix
methods.
CHAPTER 5

CONCLUSIONS, DISCUSSION, RECOMMENDATIONS AND FUTURE RESEARCH

5.1 Multiobjective Optimization of Computer Experiments

In Chapter 2 and Chapter 3, we presented a detailed exposition of the expected improvement approach to multiobjective optimization. In addition to an overview of the most current improvement function from the literature, we presented three new improvement functions (the maximin improvement function, the Gaussian-weighted hypervolume improvement, and the completeness indicator improvement function), and described in detail how one can implement all of these approaches. Additionally, a theoretical comparison of all improvement functions, based on certain desirable features was performed, as well as an empirical comparison of several combinations of improvement functions, \( QI(\cdot) \) interpretations, and dependence structures on a variety of real-world applications and test functions. Here, we will pool the collective knowledge gained from both studies to make conclusions and recommendations regarding the expected improvement approach to multiobjective optimization.

Before recommending a best approach, we will summarize some general conclusions that can be made from the empirical results in Chapter 3. First, an intuitive
The conclusion is that a larger initial design almost always leads to a better approximation to the Pareto front and set. The only example where this did not occur was the DTLZ2 function with a dependence model, which, as previously mentioned, seems to be an anomaly. Therefore, we recommend that, if possible, a design with 10 inputs per dimension, consistent with the recommendations of Loeppky et al. (2009), be used as a starting point for an expected improvement algorithm for multiobjective optimization. However, we should note that, with many of the top-performing improvement functions, the results from an initial 5 input per dimension design, while not as effective as the 10 per dimension design, performed quite well.

Second, a somewhat counterintuitive conclusion is that a nonseparable LMC dependence model offers little or no improvement over the much simpler independence model. In fact, in many examples, the dependence model actually performed worse, on average, than the independence model for several improvement functions. Additionally, when the dependence model does produce better Pareto front approximations, these approximations do not offer large gains over the independence model. The small gains that can appear do not seem to offset the increase computational demands in fitting the more complex nonseparable LMC. In fact, the only example where the dependence model seems to offer substantial gains over the independence model is the DTLZ2 example with a 20 input (5 per dimension) initial design. Specifically, the dependence model offered better approximations, in terms of the two quality indicators, for the scaling invariant approaches. For the most part, however, while the nonseparable dependence model does not perform poorly within the expected improvement framework, it does not appear to offer any real advantage over the simpler,
computationally faster independence model. Therefore, a general recommendation is to stick to with the independence model.

The third and final, broad conclusion that we come to is that neither the QIM1 or QIM2 interpretation of $QI(x)$ is consistently better. In general, it seems that there is not a clear winner when using the scaling dependent improvement functions. When using the scaling invariant improvement functions based on the completeness indicator, however, there are drastic differences. When using the **averaged** completeness indicator, ECI2 consistently produces high-quality Pareto front approximations, while ECI1 often produces low-quality approximation. As previously mentioned, ECI1 seems to push sequentially added points towards unexplored regions near the boundaries, rather than toward the true Pareto set. So, with the averaged completeness indicator, QIM2 seems superior. However, when using the **estimated** completeness indicator, EECI1 consistently produced high-quality Pareto set approximation, while EECI2 produced poor Pareto set approximations. So, with when using the estimated improvement function, QIM1 seemed superior. Based on all of these observations, one cannot make a broad recommendation as to which $QI(\cdot)$ interpretation (QIM1 or QIM2) is superior.

In addition to the general conclusions above, we now attempt to make a recommendation as to a best’ approach among all expected improvement functions (see Table 3.1 for a list of the possible choices). Judged solely on the quality of the Pareto front approximations created, methods based on the maximin fitness function (EMMI, EMAX1, EMMI2) and the hypervolume improvement function (EHI1 and EHI2) with independent Gaussian process models for each output created the best approximations of the Pareto front and set in most examples and applications. In almost every
example presented, these 5 expected improvement interpretations, along with a inde-
pendence model, yielded Pareto front and set approximations with consistently high
hypervolume indicators and consistently low binary-\(\epsilon\) indicators. Additionally, when
graphical methods could be used, these approaches created approximations that were
visually appealing, in that they were on or close to the true Pareto objectives.

Taking into account other factors, however, we recommend the maximin fitness
based methods EMMI-Ind and EMMI2-Ind. The advantage to using a sequential
design algorithm based on these two approaches over EMAX1-Ind is mainly a theo-
retical one based on the results summarized in Table 2.1. Specifically, the maximin
improvement function \(I_M(\cdot)\) has a relationship to the binary-\(\epsilon\) indicator, while \(I_P(\cdot)\)
does not appear to have a relationship to any quality indicator, and the maximin
improvement function generalizes the popular single-objective improvement function
\(I(x)\) of Jones et al. (1998) and Schonlau (1997) (in the since that \(I(x) = I_M(x)\) when
\(m = 1\)) while the Pareto improvement function \(I_P(\cdot)\) does not.

The recommendation of EMMI-Ind and EMMI2-Ind over the hypervolume based
approaches, however, is both based on theoretical and practical considerations. First,
it is well-known in the MOEA literature that for even moderately sized \(m, I_H(\cdot, \cdot),\) and
thus \(I_H(y(x)),\) requires considerable computational overhead. (This was also pointed
out in Chapter 2.) Therefore, particularly in a QIM1 setting, the creation of a sample
average approximation either requires prohibitively long hypervolume calculations,
or Monte Carlo approximations of these hypervolumes, which adds an extra layer
of random error into the sample average approximation. Secondly, \(I_H(y(x))\) requires
some additional information, as the user is required to supply a dominated point \(R.\) If
the objective function are truly black box functions, \(R\) could be difficult to identify.
Moreover, even if one can specify an upper bound on all objectives, the value of $I_H(y(x))$ will depend on particular choice of the upper bound. So, while $I_M(y(x))$ can be computed exactly for any $m$ and is free from external, user-supplied parameters, $I_H(y(x))$ is not. In Section 3.2.2, this second, somewhat theoretical limitation (in conjunction with the dependence on scaling) created practical difficulties when trying to solve the constrained multiobjective Nowacki beam problem (see Figure 3.83).

This is not to say that EMMI and EMMI2 are not without weaknesses. The main shortcoming with these improvement criteria (as well as the criteria based on $I_P$, $I_K$, and $I_W$) is that they are dependent upon the scaling of the output. This means that the user must either scale the various output to similar ranges based on prior knowledge of their ranges, or one must somehow empirically scale the outputs based on observed data. In Chapter 3, the examples used the latter approach. In Section 3.2.2 (and, more specifically, Figure 3.83), it was shown that different, seemingly reasonable empirical scaling strategies can lead to quite different results in a constrained multiobjective optimization problem. Therefore, a decision-maker could be apprehensive about using any scaling dependent approach.

If a scaling invariant improvement criteria is desired, ECI2-Ind and EECI1-Ind are recommended. On a variety of problems, these approaches, based on the completeness indicator, produced consistently better results than the other scaling invariant approaches. In fact, in some cases, the Pareto set and front approximations were actually as good or better than many of the scaling dependent methods. PI, ECI1, and EECI2, with either an independence or dependence model, should be avoided, as
these approaches, with only a few exceptions, almost always produced the worst approximations. ECI1 is particularly poor because, not only does it produce poor Pareto set approximations, but it also is the most computationally demanding approach.

While the thesis describes many new ways on can perform sequential multiobjective optimization for multiple output computer experiments and compares them to several previously proposed approaches, there are several aspects of this topic where interesting open problems remain. We will now describe three future directions one could take regarding sequential designs for the multiobjective optimization of computer experiments.

Sequential Multiobjective Optimization via Particle Learning

Suppose that the GP model $Y(x)$ for the $m$-variate function $Y(x)$ is assumed to have the form

$$Y(x) = F(x)\beta + AZ(x)$$

where $A$ is a symmetric $m \times m$ positive-definite matrix,

$$\beta = (\beta_1 \ldots \beta_m)^T,$$  \hspace{1cm} (5.1.1)

and $Z(x) = (Z_1(x), \ldots, Z_m(x))^T$ is an $m \times 1$ vector of mutually independent stationary Gaussian processes with zero mean and unit variance. Thus far, we have estimated the covariance parameters $\theta$ and $\sigma_0$ (where $\sigma_0$ is just the vector of the covariance parameters that comprise $\Sigma_0 = Cov(Y(x), Y(x)) = AA$ and $\theta$ are the within-output correlation parameters of the $Z_i(x)$ processes) with maximum likelihood or restricted maximum likelihood, effectively ignoring parameter uncertainty and it’s effect on the posterior predictive distribution of $Y(x) | Y^{n,m} = y^{n,m}$. In theory, we could put a prior on $\sigma_0$ and $\theta$, use the Metropolis-Hastings algorithm to get
draws \( \{(\sigma_0, \theta)_n^{(i)}\}_{i=1}^{Nmc} \) from their posterior distribution and (in the QIM1 case, for any improvement function \( I^*(\cdot) \)) maximize the approximated expected improvement
\[
\frac{1}{Nmc} \sum_{i=1}^{Nmc} E \left[ I^* (Y(x)) \mid Y^{n,m} = y^{n,m}, (\sigma_0, \theta)^{(i)} \right]
\]
to choose our next sequentially added design point \( x_{n+1} \). Unfortunately, this would be extremely computationally demanding, since we would need to re-fit the model via MCMC for every single sequentially added design point. It is this reason that we have used a “plug-in” estimator for the covariance parameters. However, recent research in Gramacy and Polson (2011) describes how one can use particle learning to implement a sequential Monte Carlo algorithm to quickly update the posterior at each stage in our expected improvement algorithm. Although the developments in this new paper focus on optimizing a function with a single-output in the expected improvement framework, one could easily, at least in principle, adapt the methods for multiobjective optimization in the expected improvement framework.

The model would be similar to the model described in Section 1.3, except now we will assume that \( \sigma_0 \) and \( \theta \) have prior distribution \( p(\sigma_0) \) and \( p(\theta) \), with the noninformative prior remaining on \( \beta \). First, we obtain, via Metropolis-Hastings and the Gibbs sampler, draws \( \{(\sigma_0, \theta)_n^{(i)}\}_{i=1}^{Nmc} \) from the posterior distribution of \( \sigma_0 \) and \( \theta \) given the initial data, denoted \( p(\sigma_0, \theta | y^{n,m}) \). With these draws, use some sort of direct search algorithm (e.g., MADS) to find
\[
x_{n+1} = \arg \max \frac{1}{Nmc} \sum_{i=1}^{Nmc} E \left[ I^* (Y(x)) \mid Y^{n,m} = y^{n,m}, (\sigma_0, \theta)^{(i)} \right]
\]
(5.1.2)

Next, evaluate \( y(x_{n+1}) \). Recall, in Section 1.3, we have derived the distribution of \( Y(x) \mid Y^{n,m} = y^{n,m}, \sigma_0, \theta \). Denote its p.d.f. by \( p(y(x) \mid y^{n,m}, \sigma_0, \theta) \). Rather than setting up a brand new MCMC procedure to get draws from \( p(\sigma_0, \theta | y^{n+1,m}) \), resample
with replacement from \( \{(\sigma_0, \theta)_n^{(i)}\}_{i=1}^{Nmc} \), where each \((\sigma_0, \theta)^{(i)}\) has probability \(w_n^i\) of being resampled. Here, each

\[
w_n^i \propto p(y(x_{n+1})\mid y^{n,m}, (\sigma_0, \theta)^{(i)}).
\]

Denote these newly resampled draws as \( \{(\sigma_0, \theta)^{(i)}\}_{i=1}^{Nmc} \). These draws represent a discrete approximation to \( p(\sigma_0, \theta|y^{n+1,m}) \). Using these draws, we can once again approximate the expected improvement, maximize it, and find our next sequential design point. This process can then be repeated until we have exhausted our budget, or have a satisfactory approximation of the Pareto front. The advantage here is that we only need to run our Metropolis-Hastings and Gibbs sampler once. The rest of the posterior parameter draws come from resampling the draws from the previous stage.

Of course, the difficulty here will come in the implementation. One computational difficulty that would arise is that, except in some limited biobjective cases, there are not analytic expressions for the expected improvement. Therefore,

\[
E \left[ I^* (Y(x)) \mid Y^{n,m} = y^{n,m}, (\sigma_0, \theta)^{(i)} \right]
\]

itself need to be approximated in within (5.1.2) in most cases. The other main issue regarding the implementation is the choice of parameterization and priors. For the independence case, models and parameterizations found in Higdon et al. (2008) would probably work well. Specifically, the covariance of each output could have an inverse gamma prior, and the correlation parameters could be reparameterized as

\[
\rho_i = \exp \left\{ -\frac{1}{4} \theta_i \right\}
\]

and assumed to follow a beta distribution. In the nonseparable LMC model with \( A \) symmetric and positive-definite, the same beta priors could be assumed on the
transformed $\theta$, but the challenge would be parameterizing $A$ in a nice way and eliciting priors on these covariance parameters. Some sort of parameterization that would allow some of the parameters to be on a compact intervals, such as a spherical parameterization or a givens angle parameterization, as described in Pinheiro and Bates (1996) and Daniels and Kass (1999), could be useful. Or, in the spirit of Gelfand et al. (2004), one could place an inverse-Wishart prior on $AA = \Sigma_0$.

**Updating the Design in Batches**

One possible generalization of the expected improvement algorithm for multiobjective optimization involves adding $h$ design points ($h > 1$)

$$X = (x_{B,1}, \ldots, x_{B,h})$$

at each iteration of the algorithm instead of just one design point. Schonlau (1997) and Loeppky et al. (2010) both discuss the advantages of so-called *batch-sequential designs* and how such designs can be implemented. In principal, this could be accomplished in a straightforward manner with a Pareto set approximation quality indicator. For example, if the hypervolume indicator is my quality indicator of choice, we could define a batch hypervolume improvement function (for some dominated reference point $R$) as

$$I^B_H(y(x_{B,1}), \ldots, y(x_{B,h}), R)$$

= 0 if any $\{y(x_{B,i})\} \leq \mathcal{P}_Y^n$ or $y(x_{B,i}) > y(x_{B,j})$ for some $i \neq j$

and

$$I^B_H(y(x_{B,1}), \ldots, y(x_{B,h}))$$

= $I_H(\{y(x_{B,1}), \ldots, y(x_{B,1})\} \cup \mathcal{P}_Y^n, R) - I_H(\mathcal{P}_Y^n, R)$, otherwise

285
(The above definition says we have zero improvement if any of the proposed inputs in $X_B$ dominate each other, are dominated by the current Pareto front, or do not dominate the reference point.) Then, in theory, we could try to find $x_B$ such that

$$E \left[ I_B^B(y(x_{B,1}), \ldots, y(x_{B,h})) \mid Y^{n,m} = y^{n,m} \right]$$

is maximized.

We could find the optimal batch of points by solving a constrained optimization problem. Specifically, the next set of $h$ inputs can be found by solving

$$\max E \left[ I_B^B(y(x_{B,1}), \ldots, y(x_{B,h})) \mid Y^{n,m} = y^{n,m} \right]$$

subject to $x_{B,1} \in X, \ldots, x_{B,h} \in X$.

While this is simple in principle, the real challenge would be finding an effective and efficient method to solve this rather difficult constrained optimization problem. One could possibly use some sort of numerical optimization algorithm (MADS, genetic algorithm, etc.) to obtain a solution.

**Multiobjective Optimization in the Presence of Environmental Variables**

Single-objective optimization in the presence of environmental variables (sometimes referred to as noise variables), as described in Williams et al. (2000) and Williams et al. (2010), can be useful if some inputs of a computer experiment cannot be controlled in the physical process being modeled by the computer code but have values governed by some probability distribution. In this single-objective approach, the goal is to find control variables so that the *averaged* or *integrated* output (with respect to the distribution on the environmental variables) is minimized. One could conceivably adapt an expected improvement algorithm for multiobjective optimization to approximate the Pareto front of several integrated response functions. In this
case, I would want to find the Pareto front of the functions

\[ M_i(x_c) = \sum_{j=1}^{n_e} w_j y_i(x_c, x_{e,j}) \]

for \( i = 1, \ldots, m \). Here, \( x_c \) are control variables and \( x_e \) are environmental variables, which take can take on values \( x_{e,1}, \ldots, x_{e,n_e} \) with probabilities \( w_1, \ldots, w_{n_e} \). This is an interesting topic, because all expected improvement algorithms for multiobjective optimization in the literature treat all inputs as control variables, and the issue of control vs. environmental variables is essentially ignored.

## 5.2 Sensitivity Analysis of Computer Experiments

In Chapter 4, we have shown how to estimate sensitivity indices in a Gaussian process model in a hybrid Bayesian/plug-in manner. Our computations allow for the GP to have any mean that is a polynomial function of the inputs, and they also allow for the GP to have an anisotropic product correlation structure with either a Gaussian, Cubic, or Bohman correlation function. We have shown that a polynomial mean structure can lead to more accurate estimates of sensitivity indices in situations where the initial design is small and a non-constant trend is known to exist. Additionally, we have also shown that a polynomial mean, in conjunction with a compactly supported correlation function (such as the Bohman or the cubic), a restricted correlation parameter space, and sparse matrix methods can allow for effective and more efficient estimation of sensitivity indices when the initial design of a computer experiment is large.

As we pointed out in the Chapter 4, the sensitivity indices in the examples were calculated by plugging in likelihood-based estimates for these unknown parameters.
Future research efforts could possibly improve upon our results by calculating sensitivity indices in fully Bayesian manner. More specifically, on could place a prior distribution \([\beta, \lambda_Y, \lambda_e, \theta]\) on the unknown parameters and estimate the necessary integrals as

\[
\hat{V}_S^j = \mathbb{E} \{ \mathbb{E} \{ \text{Var} \{ g(Y(X)|X_S) \} | \beta, \lambda_Y, \lambda_e, \theta, Z_{\text{sim}} \} | Z_{\text{sim}} \},
\]

where the outer expectation is taken with respect to the posterior distribution of the parameters. Such estimation is attractive, as it would take parameter uncertainty into account.

Sensitivity indices and their estimates in the Gaussian process framework, as presented in Chapter 1 and Chapter 4, were based on the underlying assumption that the input space \(\mathcal{X}\) is a \(d\)-dimensional rectangle. However, many real-world computer experiments, the input space is not a hypercube. For example, many practical examples have linearly constrained input regions of the form

\[
\mathcal{X}_c = \{ \mathbf{x} : \mathbf{A} \mathbf{x} \leq \mathbf{b}, \ l_i \leq x_i \leq u_i \forall i \in \{1, \ldots, d\} \},
\]

(5.2.1)

where \(\mathbf{A}\) is an \(f \times d\) matrix and \(\mathbf{b}\) is a \(f \times 1\) vector for some positive integer \(f\). While the Sobol’ decomposition fails if we assume \(\mathbf{X}\) is uniformly distributed over the set (5.2.1), one can still define the sensitivity indices in terms of joint effect variances of the form

\[
V_S^j = \text{Var}_{g^*} \{ \mathbb{E}_{g^*}[g(\mathbf{X})|X_S] \},
\]

where \(g^*(\cdot)\) is the probability density function of a random vector uniformly distributed over (5.2.1), i.e,

\[
g^*(\mathbf{x}) = \frac{1}_{\mathcal{X}_c} = \frac{1}_{\mathcal{X}_c} \int_{\mathcal{X}_c} d\mathbf{x}.
\]
The challenge here would be in calculating the necessary integrals

\[ \hat{V}_S^j = E_p \{ Var_g^* \{ E_g^* \{ Y(X) | X_S \} | Z_{sim} \} \} \]  

in the Gaussian process framework. Recall that our derivations of the joint effect variances in Chapter 4 relied heavily on the assumption that the input distributions were independent and that the input region was rectangular. Successful calculation and implementation of the integrals of the form (5.2.2) could further increase the class of problems for which sensitivity indices can be calculated.
APPENDIX A

MULTIVARIATE MODELING DERIVATIONS AND PROOFS

A.1 Proof of Theorem 1

We begin by stating two lemmas.

Lemma 1. For any $n \times 1$ vector $v$ and any $n \times n$ symmetric, positive definite matrix $A$,

$$\int_{\mathbb{R}^n} \exp \left\{ -\frac{1}{2} w^\top A^{-1} w + v^\top w \right\} dw = (2\pi)^{n/2} |A|^{1/2} \exp \left\{ \frac{1}{2} v^\top Av \right\}.$$

(A useful interpretation of this result is that if $W$ has density $f(w)$ for which

$$f(w) \propto \exp \left\{ -\frac{1}{2} w^\top A^{-1} w + v^\top w \right\},$$

then $W \sim N_n [Av, A]$.)

Proof. See Lemma B.1.1 in Santner et al. (2003). \hfill \Box

Lemma 2. Suppose that $B$ is a nonsingular $n \times n$ matrix, $C$ is nonsingular $m \times m$ matrix, and $A$ is an arbitrary $n \times m$ matrix such that $(A^\top B^{-1} A + C)^{-1}$ is nonsingular.

Then $(B + AC^{-1} A^\top)$ is an $n \times n$ nonsingular matrix with inverse given by

$$(B + AC^{-1} A^\top)^{-1} = B^{-1} - B^{-1} A (A^\top B^{-1} A + C)^{-1} A^\top B^{-1}.$$
Proof. See Lemma B.3.2 in Santner et al. (2003).

Via standard multivariate normal results, we have

\[ Y_1 | Y_2 = y_2, \beta \]  \hspace{1cm} (A.1.1)

\[ \sim N \left( F_1 \beta + \Sigma_{12} \Sigma_{22} (y_2 - F_2 \beta), \Sigma_{11} - \Sigma_{12} \Sigma_{22}^{-1} \Sigma_{12}^\top \right). \]  \hspace{1cm} (A.1.2)

Notice that

\[ f (y_1 | y_2) \propto \int_{\mathbb{R}^p} f (y_1, y_2, \beta) \, d\beta \]

\[ = \int_{\mathbb{R}^p} f (y_1 | y_2, \beta) f (y_2 | \beta) f (\beta) \, d\beta \]  \hspace{1cm} (A.1.3)

Therefore, to prove the theorem, it suffices to show that (A.1.3) is proportional to a multivariate normal pdf with mean \( \mu_{12} \) and covariance matrix \( \Sigma_{12} \). We have

\[ f (y_1 | y_2) \propto \int_{\mathbb{R}^p} \exp \left\{ -\frac{1}{2} (y_1 - F_1 \beta - \Sigma_{12} \Sigma_{22}^{-1} (y_2 - F_2 \beta))^\top \right. \]

\[ \times \left( \Sigma_{11} - \Sigma_{12} \Sigma_{22}^{-1} \Sigma_{12}^\top \right)^{-1} \]

\[ \times (y_1 - F_1 \beta - \Sigma_{12} \Sigma_{22}^{-1} (y_2 - F_2 \beta) - \frac{1}{2} (y_2 - F_2 \beta)^\top \Sigma_{22}^{-1} (y_2 - F_2 \beta)) \}

\[ \times \left\{ (V^{-1} \Sigma_{12} \Sigma_{22}^{-1} y_2)^\top y_1 - \frac{1}{2} y_1^\top V^{-1} y_1 \right\} \]

\[ \times \int_{\mathbb{R}^p} \exp \left\{ -\frac{1}{2} \beta^\top \left( F_2^\top \Sigma_{22}^{-1} F_2 + D^\top V^{-1} D \right) \beta \right. \]

\[ + (F_2^\top \Sigma_{22}^{-1} y_2 + D^\top V^{-1} (y_1 - \Sigma_{12} \Sigma_{22}^{-1} y_2))^\top \beta \}

\[ d\beta, \]

where \( D = F_1 - \Sigma_{12} \Sigma_{22}^{-1} F_2 \) and \( V = \Sigma_{11} - \Sigma_{12} \Sigma_{22}^{-1} \Sigma_{12}^\top \). Applying Lemma 1 with

\[ A = \left( F_2^\top \Sigma_{22}^{-1} F_2 + D^\top V^{-1} D \right)^{-1} \]

and

\[ v = \left[ F_2^\top \Sigma_{22}^{-1} y_2 + D^\top V^{-1} (y_1 - \Sigma_{12} \Sigma_{22}^{-1} y_2) \right]^\top \]
yields

\[ f ( y_1 | y_2 ) \]

\[ \propto \exp \left\{ \left( \begin{array}{c} V^{-1} \Sigma_{12} \Sigma_{22}^{-1} y_2 \\ - \frac{1}{2} y_1^\top V^{-1} y_1 \end{array} \right) \right\} \]

\[ \times \exp \left\{ \frac{1}{2} \left( F_2^\top \Sigma_{22} y_2 + D^\top V^{-1} ( y_1 - \Sigma_{12} \Sigma_{22}^{-1} y_2 ) \right) \right\} \]

\[ \times \left( F_2^\top \Sigma_{22}^{-1} F_2 + D^\top V^{-1} D \right)^{-1} \]

\[ \times \left( F_2^\top \Sigma_{22}^{-1} y_2 + D^\top V^{-1} ( y_1 - \Sigma_{12} \Sigma_{22}^{-1} y_2 ) \right) \} \]

\[ \propto \exp \left\{ - \frac{1}{2} y_1^\top \left[ V^{-1} - V^{-1} D \left( F_2^\top \Sigma_{22}^{-1} F_2 + D^\top V^{-1} D \right)^{-1} D^\top V^{-1} \right] y_1 \right\} \]

\[ + \left[ ( V^{-1} \Sigma_{12} \Sigma_{22}^{-1} y_2 ) \right] \]

\[ + \left( F_2^\top \Sigma_{22}^{-1} y_2 - D^\top V^{-1} \Sigma_{12} \Sigma_{22}^{-1} y_2 \right) \left( F_2^\top \Sigma_{22}^{-1} F_2 + D^\top V^{-1} D \right)^{-1} D^\top V^{-1} \}

Letting

\[ A_0 = \left( V^{-1} - V^{-1} D \left( F_2^\top \Sigma_{22}^{-1} F_2 + D^\top V^{-1} D \right)^{-1} D^\top V^{-1} \right)^{-1} \]

and

\[ v_0 \]

\[ = \left[ ( V^{-1} \Sigma_{12} \Sigma_{22}^{-1} y_2 ) \right] \]

\[ + \left( F_2^\top \Sigma_{22}^{-1} y_2 - D^\top V^{-1} \Sigma_{12} \Sigma_{22}^{-1} y_2 \right) \left( F_2^\top \Sigma_{22}^{-1} F_2 + D^\top V^{-1} D \right)^{-1} D^\top V^{-1} \]

we can again apply Lemma 1 with \( A = A_0 \) and \( v = v_0 \) to conclude that \( Y_1 | Y_2 = y_2 \sim N ( A_0 v_0, A_0 ) \).

To complete the proof, we must show that \( A_0 = \Sigma_{1|2} \) and \( A_0 v_0 = \mu_{1|2} \). We can apply Lemma 2 to show

\[ A_0 = V - V ( V^{-1} D ) \left[ ( D^\top V^{-1} ) V ( V^{-1} D ) \right] \]
\[
- \left( \mathcal{F}_2^\top \Sigma_{22}^{-1} \mathcal{F}_2 + D^\top V^{-1} D \right)^{-1} (D^\top V^{-1}) V \\
= V + D \left( \mathcal{F}_2^\top \Sigma_{22}^{-1} \mathcal{F}_2 \right)^{-1} D^\top \\
= \Sigma_{11} - \Sigma_{12} \Sigma_{22}^{-1} \Sigma_{12}^\top \\
+ (\mathcal{F}_1 - \Sigma_{12} \Sigma_{22}^{-1} \mathcal{F}_2) (\mathcal{F}_2^\top \Sigma_{22}^{-1} \mathcal{F}_2)^{-1} (\mathcal{F}_1 - \Sigma_{12} \Sigma_{22}^{-1} \mathcal{F}_2)^\top \\
= \Sigma_{1|2}.
\]

Finally,

\[
A_0 v_0 \\
= \left[ V + D \left( \mathcal{F}_2^\top \Sigma_{22}^{-1} \mathcal{F}_2 \right)^{-1} D^\top \right] \\
\times \left[ (V^{-1} \Sigma_{12} \Sigma_{22}^{-1} y_2)^\top \\
+ (\mathcal{F}_2^\top \Sigma_{22}^{-1} y_2 - D^\top V^{-1} \Sigma_{12} \Sigma_{22}^{-1} y_2) (\mathcal{F}_2^\top \Sigma_{22}^{-1} \mathcal{F}_2 + D^\top V^{-1} D)^{-1} D^\top V^{-1} \right]^\top \\
= \{ \Sigma_{12} \Sigma_{22}^{-1} \\
+ D \left( \mathcal{F}_2^\top \Sigma_{22}^{-1} \mathcal{F}_2 + D^\top V^{-1} D \right)^{-1} (\mathcal{F}^\top \Sigma_{22}^{-1} - D^\top V^{-1} \Sigma_{12} \Sigma_{22}^{-1}) \\
+ D \left( \mathcal{F}_2^\top \Sigma_{22}^{-1} \mathcal{F}_2 \right)^{-1} D^\top \left[ V^{-1} \Sigma_{12} \Sigma_{22}^{-1} \right] \\
+ V^{-1} D \left( \mathcal{F}_2^\top \Sigma_{22}^{-1} \mathcal{F}_2 + D^\top V^{-1} D \right)^{-1} \\
\times (\mathcal{F}^\top \Sigma_{22}^{-1} \mathcal{F} - D^\top V^{-1} \Sigma_{12} \Sigma_{22}^{-1}) \} \} \} y_2 \\
= \Sigma_{12} \Sigma_{22}^{-1} y_2 + D \left( \mathcal{F}_2^\top \Sigma_{22}^{-1} \mathcal{F}_2 \right)^{-1} D^\top V^{-1} \Sigma_{12} \Sigma_{22}^{-1} y_2 \\
+ \left[ D + D \left( \mathcal{F}_2^\top \Sigma_{22}^{-1} \mathcal{F}_2 \right)^{-1} D^\top V^{-1} D \right] \\
\times (\mathcal{F}_2^\top \Sigma_{22}^{-1} \mathcal{F}_2 + D^\top V^{-1} D)^{-1} \\
\times (\mathcal{F}^\top \Sigma_{22}^{-1} - D^\top V^{-1} \Sigma_{12} \Sigma_{22}^{-1}) \} \} y_2 \\
= \Sigma_{12} \Sigma_{22}^{-1} y_2 \\
+ D \left( \mathcal{F}_2^\top \Sigma_{22}^{-1} \mathcal{F}_2 \right)^{-1} D^\top V^{-1} \Sigma_{12} \Sigma_{22}^{-1} y_2
\]
A.2 Proof of Theorem 2

To begin, we first reorder the observations at $\mathcal{D}_n$ as

$$y_{m,n}^* = (y_1(x_1), \cdots, y_1(x_n), \cdots, y_m(x_1), \cdots, y_m(x_n)) = (y_1^0, \cdots, y_m^0)$$

and reorder $Y^0$ as

$$Y_*^0 = (Y_1(x_1^0), \cdots, Y_1(x_{n_0}^0), \cdots, Y_m(x_1^0), \cdots, Y_m(x_{n_0}^0)) = (Y_1^0, \cdots, Y_m^0).$$

In this case,

$$\begin{bmatrix} Y_*^0 \\ Y_{m,n}^* \end{bmatrix} | \beta \sim \mathcal{N} \left( \begin{bmatrix} F_n^* \\ \Sigma_{0,n_0,n}^* \end{bmatrix} \beta, \begin{bmatrix} \Sigma_0^* & \Sigma_{0,n_0,n}^* \\ \Sigma_{0,n_0,n}^* & \Sigma_{m,n}^* \end{bmatrix} \right)$$

where

$$F_n^* = \begin{bmatrix} F_1^n & 0 & \cdots & 0 \\ 0 & F_2^n & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & F_m^n \end{bmatrix},$$

$$F_i^n = [f_i(x_1) \cdots f_i(x_n)]^\top,$$
\[
\mathcal{F}_0^* = \begin{bmatrix}
F_1^0 & 0 & \cdots & 0 \\
0 & F_2^0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & F_m^0
\end{bmatrix},
\]

\[
F_m^0 = [f_i(x_1^0) \cdots f_i(x_m^0)]^\top,
\]

\[
\Sigma_{m,n}^* = \begin{bmatrix}
\text{cov} (y_1^n, y_1^n) & \text{cov} (y_1^n, y_2^n) & \cdots & \text{cov} (y_1^n, y_m^n) \\
\text{cov} (y_2^n, y_1^n) & \text{cov} (y_2^n, y_2^n) & \cdots & \text{cov} (y_2^n, y_m^n) \\
\vdots & \vdots & \ddots & \vdots \\
\text{cov} (y_m^n, y_1^n) & \text{cov} (y_m^n, y_2^n) & \cdots & \text{cov} (y_m^n, y_m^n)
\end{bmatrix},
\]

\[
\Sigma_{0,n_0,n}^* = \begin{bmatrix}
\text{cov} (y_0^0, y_1^n) & \text{cov} (y_0^0, y_2^n) & \cdots & \text{cov} (y_0^0, y_m^n) \\
\text{cov} (y_0^0, y_1^n) & \text{cov} (y_0^0, y_2^n) & \cdots & \text{cov} (y_0^0, y_m^n) \\
\vdots & \vdots & \ddots & \vdots \\
\text{cov} (y_0^0, y_1^n) & \text{cov} (y_0^0, y_2^n) & \cdots & \text{cov} (y_0^0, y_m^n)
\end{bmatrix}
\]

and

\[
\Sigma_y^0 = \begin{bmatrix}
\text{cov} (y_1^n, y_1^n) & \text{cov} (y_2^n, y_2^n) & \cdots & \text{cov} (y_1^n, y_m^n) \\
\text{cov} (y_2^n, y_2^n) & \text{cov} (y_2^n, y_2^n) & \cdots & \text{cov} (y_2^n, y_m^n) \\
\vdots & \vdots & \ddots & \vdots \\
\text{cov} (y_m^n, y_2^n) & \text{cov} (y_m^n, y_2^n) & \cdots & \text{cov} (y_m^n, y_m^n)
\end{bmatrix}
\]

Under this particular ordering, we can use Theorem 1 to show that
\[ Y^*_m | Y^*_m = y^*_m \sim N(y^*_m, \Sigma^0_y) \]

where

\[
\hat{y}_m^* = \mathcal{F}_n^{*\top} \hat{\beta}^* + \Sigma_{0,n_0,n}^* \Sigma_{m,n}^{*\top} (y_m^{m,n} - \mathcal{F}_n^{*\top} \hat{\beta}^*),
\]

\[
S_m^0 = \Sigma_y^0 - \Sigma_{0,n_0,n}^* \Sigma_{m,n}^{*\top} \Sigma_{0,n_0,n}^{\top}
\]

\[
+ (\mathcal{F}_n^{*\top} \Sigma_{0,n_0,n}^{*\top} \mathcal{F}_n^{*\top}) (\mathcal{F}_n^{*\top} \Sigma_{m,n}^{*\top} \mathcal{F}_n^{*\top})^{-1} (\mathcal{F}_n^{*\top} - \Sigma_{0,n_0,n}^* \Sigma_{m,n}^{*\top} \mathcal{F}_n^{*\top})^{\top}
\]

and

\[
\hat{\beta}^* = (\mathcal{F}_n^{*\top} \Sigma_{m,n}^{*\top} \mathcal{F}_n^{*\top})^{-1} \mathcal{F}_n^{*\top} \Sigma_{m,n}^{*\top} y_m^{m,n}.
\]

Under the assumption that the components \(W_1(x), \ldots, W_m(x)\) of \(W(x)\) are independent stationary GaSPs with covariance functions \(\sigma^2_i R_i(\cdot; \theta_i)\), we have

\[
\Sigma_{m,n}^* = \text{blkdiag} (\text{cov} (Y_1^n, Y_1^n), \cdots, \text{cov} (Y_m^n, Y_m^n)) = \text{blkdiag} (\sigma^2_1 R_1, \cdots, \sigma^2_m R_m),
\]

295
\[ \Sigma_0^0 = \text{blkdiag} \left( \text{cov} \left( \mathbf{Y}_1^0, \mathbf{Y}_1^0 \right), \ldots, \text{cov} \left( \mathbf{Y}_m^0, \mathbf{Y}_m^0 \right) \right) = \text{blkdiag} \left( \sigma_1^2 I_n, \ldots, \sigma_m^2 I_n \right), \]

and

\[ \Sigma^*_{0,n_0,n} = \text{blkdiag} \left( \text{cov} \left( \mathbf{Y}_1^0, \mathbf{Y}_1^n \right), \ldots, \text{cov} \left( \mathbf{Y}_m^0, \mathbf{Y}_m^n \right) \right) = \text{blkdiag} \left( \sigma_1^2 \mathbf{r}_1^\top, \ldots, \sigma_m^2 \mathbf{r}_m^\top \right), \]

where \( \mathbf{R}_k \) is an \( n \times n \) matrix with \( \mathbf{R}_{ki} = \mathbf{R}_k (\mathbf{x}_i - \mathbf{x}_j; \theta_k) \) and \( \mathbf{r}_k \) is an \( n \times n_0 \) matrix with \( \mathbf{r}_k = [\mathbf{r}_k(\mathbf{x}_i^0), \ldots, \mathbf{r}_k(\mathbf{x}_n^0)] \), where each

\[ \mathbf{r}_k(\mathbf{x}_i^0) = \left[ \mathbf{R}_k (\mathbf{x}_1 - \mathbf{x}_i^0 \theta_k), \ldots, \mathbf{R}_k (\mathbf{x}_n - \mathbf{x}_i^0 \theta_k) \right]^\top. \]

Under the block diagonal structure, we have \( \mathbf{\hat{y}}_i^0 = (\mathbf{y}_1^0, \ldots, \mathbf{y}_m^0) \) and

\[ \mathbf{S}_i^0 = \text{blkdiag} \left[ \sigma_1^2 \mathbf{R}_1^0, \ldots, \sigma_m^2 \mathbf{R}_m^0 \right], \]

where

\[ \mathbf{y}_i^0 = \mathbf{F}_i^0 \mathbf{\hat{\beta}}_i + \mathbf{r}_i^\top \mathbf{R}_i^{-1} \left( \mathbf{y}_i^n - \mathbf{F}_i^0 \mathbf{\hat{\beta}}_i \right), \quad (A.2.1) \]

\[ \mathbf{R}_i^0 = \mathbf{I}_n - \mathbf{r}_i^\top \mathbf{R}_i^{-1} \mathbf{r}_i 
+ \left( \mathbf{F}_i^0 - \mathbf{r}_i^\top \mathbf{R}_i^{-1} \mathbf{F}_i^0 \right) \left( \mathbf{F}_i^n \mathbf{R}_i^{-1} \mathbf{F}_i^m \right)^{-1} \left( \mathbf{F}_i^0 - \mathbf{r}_i^\top \mathbf{R}_i^{-1} \mathbf{F}_i^m \right)^\top \quad (A.2.2) \]

and

\[ \mathbf{\hat{\beta}}_i = (\mathbf{F}_i^0 - \mathbf{r}_i^\top \mathbf{R}_i^{-1} \mathbf{F}_i^0) \left( \mathbf{F}_i^n \mathbf{R}_i^{-1} \mathbf{F}_i^m \right)^{-1} \mathbf{F}_i^n \mathbf{R}_i^{-1} \mathbf{y}_i^0. \quad (A.2.3) \]

Finally, the block diagonal structure of \( \mathbf{S}_i^0 \) implies that the covariance of \( \mathbf{Y}_i(\mathbf{x}_k^0) \) and \( \mathbf{Y}_j(\mathbf{x}_l^0) \), conditional on the observations at \( \mathcal{D}_n \), can only be nonzero if \( i = j \).

**A.3  Proof that \( \mathbf{\hat{y}}(\mathbf{x}_0) \) and \( \mathbf{S}(\mathbf{x}_0) \) are equivalent to the BLUP and MSPE matrix when \( n_0 = 1 \)**

Here, we will introduce the concept of best linear unbiased prediction of multivariate functions using the same ordering of the observations at \( \mathcal{D}_n \) as Appendix A.2,
since this ordering was used in Ver Hoef and Cressie (1993), our main reference for optimal multivariable spatial prediction. In this setting, let

\[ \Sigma_{0,n}^* \]

\[ = \text{cov} \left( Y(x_0), Y_{m,n}^* \right) = \left[ \text{cov} \left( Y(x_0), Y_{1,n}^* \right), \text{cov} \left( Y(x_0), Y_{2,n}^* \right), \ldots, \text{cov} \left( Y(x_0), Y_{m,n}^* \right) \right] \]

and recall that

\[ \Sigma_0 = \text{cov} \left( Y(x_0), Y(x_0) \right) \]

and

\[ E[Y(x_0)] = F(x_0) \beta. \]

In the multivariate case, linear unbiased predictors must be of the form \( B' y_{m,n}^* \) and must satisfy \( E[B' Y_{m,n}^*] = F(x_0) \beta \). Among predictors that satisfy these requirements, the optimal one is the predictor \( \hat{y}_{\text{BLUP}}(x_0) \) such that

\[ M' - M \equiv E \left\{ [y'(x_0) - y(x_0)] [y'(x_0) - y(x_0)]^\top \right\} \]

\[ - E \left\{ [\hat{y}_{\text{BLUP}}(x_0) - y(x_0)] [\hat{y}_{\text{BLUP}}(x_0) - y(x_0)]^\top \right\} \]

is always nonnegative definite for any unbiased linear predictor \( \hat{y}'(x_0) \). \( M \) is the mean square prediction error (MSPE) matrix of the \( \hat{y}_{\text{BLUP}}(x_0) \). Ver Hoef and Cressie (1993) prove that

\[ \hat{y}_{\text{BLUP}}(x_0) = F(x_0) (F_n^\top \Sigma_{m,n}^* F_n^*)^{-1} F_n^\top \Sigma_{m,n}^* y_{m,n}^* \]

\[ + \Sigma_{0,n0}^* \Sigma_{m,n}^* \left( y_{m,n}^* - (F_n^\top \Sigma_{m,n}^* F_n^*)^{-1} F_n^\top \Sigma_{m,n}^* y_{m,n}^* \right) \]

and

\[ M = \Sigma^0 - \Sigma_{0,n0}^* \Sigma_{m,n}^* \Sigma_{0,n0,n}^\top \]

\[ + (F(x_0) - \Sigma_{0,n0}^* \Sigma_{m,n}^* F_n^*) (F_n^\top \Sigma_{m,n}^* F_n^*)^{-1} F(x_0) \]

\[ - \Sigma_{0,n0}^* \Sigma_{m,n}^* F_n^\top \]

297
To prove that $\hat{y}_{\text{BLUP}}(x_0) = \hat{y}(x_0)$ and $M = S(x_0)$, first let $P$ be the permutation matrix such that $PY_0^{m,n} = y^{m,n}$. We then have

$$
\Sigma_{0,0} = \text{cov}(Y(x_0), PY_0^{m,n}) = \Sigma_{0,0} P^T,
$$

$$
F_n = E[PY_0^{m,n}] = P F_n^*,
$$

and

$$
\Sigma_{m,n} = \text{cov}(PY_0^{m,n}, PY_0^{m,n}) = P \Sigma_{m,n}^* P^T.
$$

Also, since $P$ is a permutation matrix, it must be orthogonal, so that $PP^T = P^TP = I_{mn}$ and $(P \Sigma_{m,n}^* P^T)^{-1} = P \Sigma_{m,n}^{-1} P^T$. Using these facts, we then have

$$
\hat{y}_{\text{BLUP}}(x_0)
= F(x_0) \left[ F_n^* (P^T P) \Sigma_{m,n}^{*^{-1}} (P^T P) F_n^* \right]^{-1} F_n^* (P^T P) \Sigma_{m,n}^{*^{-1}} (P^T P) y_0^{m,n}
+ \Sigma_{0,0} (P^T P) \Sigma_{m,n}^{*^{-1}} (P^T P)
\times \left[ y_0^{m,n} - (F_n^* (P^T P) \Sigma_{m,n}^{*^{-1}} (P^T P) F_n^*)^{-1} F_n^* (P^T P) \Sigma_{m,n}^{*^{-1}} (P^T P) y_0^{m,n} \right]
= F(x_0) \left[ F_n^* P^T (P \Sigma_{m,n}^* P^T)^{-1} P F_n^* \right]^{-1} F_n^* P^T (P \Sigma_{m,n}^* P^T)^{-1} P y_0^{m,n}
+ \Sigma_{0,0} P^T \left( P \Sigma_{m,n}^* P^T \right)^{-1} \left[ P y_0^{m,n} \right]
- (F_n^* P^T (P \Sigma_{m,n}^* P^T)^{-1} P F_n^*)^{-1} F_n^* P^T (P \Sigma_{m,n}^* P^T)^{-1} P y_0^{m,n}
= F(x_0) \hat{\beta} + \Sigma_{0,0} \Sigma_{m,n} (y_0^{m,n} - F_n \hat{\beta})
= \hat{y}(x_0).
$$

By a similar argument, one can easily show that $M = S(x_0)$. 

298
APPENDIX B

PROOFS OF THEOREMS RELATED TO THE EXPECTED MAXIMIN IMPROVEMENT FUNCTION

B.1 Proof of Theorem 4

Here are some facts that are useful in the proof:

1. $\mathcal{P}_y^n \succeq \mathcal{P}_y^{n+1}(y(x))$ since adding any point from the objective space will, at worst, leave our set of nondominated points unchanged. As a consequence,

$$I^+\left(\mathcal{P}_y^n, \mathcal{P}_y^{n+1}(y(x))\right) \geq 0,$$

as Zitzler et al. (2003) points out that $I^+ (A, B) < 0$ if and only if $A$ strictly dominates $B$.

2. Suppose that $y(x)$ is dominated by some point in $\mathcal{P}_y^n$. Then, $I_M(y(x)) = 0$ and $\mathcal{P}_y^n = \mathcal{P}_y^{n+1}(y(x))$.

3. Suppose that $y(x)$ is not dominated by any points in $\mathcal{P}_y^n$. Then, if we let

$$G_y = \{ y \in \mathcal{P}_y^n : y(x) \not\succ y \}$$

$$= \left\{ y \in \mathcal{P}_y^n : \max_{1 \leq k \leq m} (y_k(x) - y_k) > 0 \right\},$$
we have $\mathcal{P}^{n+1}_Y(y(x)) = \{y(x)\} \cup G_Y$. Notice that it is possibly that $G_Y$ is empty.

4. According to Zitzler et al. (2003), if $A$ and $B$ are Pareto front approximations,

$$I_e^+(A, B) = \max_{y^{(i)} \in A} \min_{z^{(i)} \in B} \max_{1 \leq k \leq m} \left( z^{(i)}_k - y^{(j)}_k \right)$$

5. According to Balling (2003), the maximin fitness function (see 2.2.17), which $I_M(\cdot)$ and $I_P(\cdot)$ are based on, has the following properties:

- fitness ($y(x)$) $\geq 0$ if $y(x)$ is dominated by any vector in $\mathcal{P}^n_Y$.
- fitness ($y(x)$) $\leq 0$ if $y(x)$ is not dominated by any vector in $\mathcal{P}^n_Y$.
- fitness ($y(x)$) $= 0$ if $y(x) \in \mathcal{P}^n_Y$.

We now can prove Theorem 4.

Proof of Theorem 4. Consider two cases:

1. Suppose $y(x)$ is dominated by some vector in $\mathcal{P}^n_Y$. Then,

$$\mathcal{P}^n_Y = \mathcal{P}^{n+1}_Y(y(x)),$$

so that

$$I_e^+ (\mathcal{P}^n_Y, \mathcal{P}^{n+1}_Y(y(x))) = I_e^+ (\mathcal{P}^n_Y, \mathcal{P}^n_Y) = 0.$$

Also, $I_M(y(x)) = 0$, so equality holds.

2. Suppose $y(x)$ is not dominated by any vector in $\mathcal{P}^n_Y$. Then, $\mathcal{P}^{n+1}_Y(y(x)) = \{y(x)\} \cup G_Y$ and

$$I_e^+ (\mathcal{P}^n_Y, \mathcal{P}^{n+1}_Y(y(x))) = \max_{y^{(i)} \in \mathcal{P}^{n+1}_Y(y(x))} \min_{z^{(i)} \in \mathcal{P}^n_Y} \max_{1 \leq k \leq m} \left( z^{(i)}_k - y^{(j)}_k \right).$$
For \( y^j \in G^Y \), we have

\[
\min_{z^{(i)} \in \mathcal{P}_Y} \max_{1 \leq k \leq m} \left( z^{(i)}_k - y^{(j)}_k \right) = - \max_{z^{(i)} \in \mathcal{P}_Y} \min_{1 \leq k \leq m} \left( y^{(j)}_k - z^{(i)}_k \right) = - \text{fitness}(y^{(j)}) = 0
\]

since \( y^{(j)} \in \mathcal{P}_Y^n \). For \( y(x) \), we have

\[
\min_{z^{(i)} \in \mathcal{P}_Y} \max_{1 \leq k \leq m} \left( z^{(i)}_k - y_k(x) \right) = - \max_{z^{(i)} \in \mathcal{P}_Y} \min_{1 \leq k \leq m} \left( y_k(x) - z^{(i)}_k \right) = - \text{fitness}(y(x)) \geq 0
\]

as \( y(x) \) is nondominated by \( \mathcal{P}_Y^n \). Therefore,

\[
\min_{z^{(i)} \in \mathcal{P}_Y} \max_{1 \leq k \leq m} \left( z^{(i)}_k - y^{(j)}_k \right)
\]

is maximized when \( y^{(j)} = y(x) \). Therefore, we have

\[
I_{e^+} (\mathcal{P}^n_Y, \mathcal{P}^{n+1}_Y (x)) = \max_{y^{(j)} \in \mathcal{P}^{n+1}_Y} \min_{z^{(i)} \in \mathcal{P}_Y} \max_{1 \leq k \leq m} \left( z^{(i)}_k - y^{(j)}_k \right) = - \max_{z^{(i)} \in \mathcal{P}^n_Y} \min_{1 \leq k \leq m} \left( y_k(x) - z^{(i)}_k \right) = - \max_{z^{(i)} \in \mathcal{P}^n_Y} \min_{1 \leq k \leq m} \left( y_k(x) - z^{(i)}_k \right) \\
\times 1 \left[ - \max_{z^{(i)} \in \mathcal{P}^n_Y} \min_{1 \leq k \leq m} \left( y_k(x) - z^{(i)}_k \right) > 0 \right] = I_M (y(x))
\]

(The indicator function above will be equal to 1 since \(- \text{fitness}(y(x)) > 0\), as \( y(x) \) is not dominated by any elements in \( \mathcal{P}_Y^n \) in this case.)

\[\square\]
B.2 Proof of Theorem 5

First, we will state and prove two intermediate results.

Result 2. Let \( a, m \) and \( \mu \) be any real numbers, \( s_1 \) and \( s_2 \) be any positive numbers, and \( \rho \in (-1, 1) \). Then,

\[
\int_{-\infty}^{m} \frac{s_1 - s_2 \rho}{\sqrt{(1 - \rho^2)s_2^2}} \phi \left( \frac{m - y - a + s_2 \rho(y - \mu)/s_1}{\sqrt{(1 - \rho^2)s_2^2}} \right) \phi \left( \frac{y - \mu}{s_1} \right) \, dy \quad (B.2.1)
\]

where

\[
v = \frac{\mu}{s_1^2} + \frac{m - a + \rho s_2 \mu / s_1}{\sqrt{(1 - \rho^2)s_2^2}},
\]

and

\[
A = \frac{(1 - \rho^2)s_2^2 s_1^2}{s_1^2 + s_2^2 - 2\rho s_1 s_2}
\]

Proof. First, if we let \( r = \frac{s_1}{s_1 - s_2 \rho} \), notice that

\[
\left( \frac{1}{s_1^2} + \frac{1}{(1 - \rho^2)s_2^2 r^2} \right)^{-1} = \frac{(1 - \rho^2)s_2^2 s_1^2 r^2}{(1 - \rho^2)s_2^2 + s_1^2}
\]

\[
= \frac{(1 - \rho^2)s_2^2 s_1^2}{(1 - \rho^2)s_2^2 + s_1^2/r^2}
\]

\[
= \frac{(1 - \rho^2)s_2^2 s_1^2}{(1 - \rho^2)s_2^2 + (s_1 - s_2 \rho)^2}
\]

\[
= \frac{(1 - \rho^2)s_2^2 s_1^2}{s_1^2 + s_2^2 - 2\rho s_1 s_2}
\]

\[
= A
\]
Then, by expanding the exponential terms in the normal densities, we can show that (B.2.1) is equal to

\[
\int_{-\infty}^{m} \exp \left\{ -\frac{1}{2} \left[ \frac{\mu^2}{s_1^2} + \frac{(m - a + \rho s_2 \mu / s_1)^2}{\sqrt{1 - \rho^2}) s_2^2} \right] \right\} \, dy
\]

Also, by completing the square, we can show that

\[
\int_{-\infty}^{m} \frac{1}{\sqrt{2\pi}} \exp \left\{ -\frac{1}{2} A^{-1} y^2 + vy \right\} \, dy = \sqrt{A} \exp \left\{ \frac{1}{2} A v^2 \right\} \phi \left( \frac{m - Av}{\sqrt{A}} \right),
\]

which yields the desired result.

\[
\text{Result 3.} \quad \text{Under the same assumptions as Result 2, we have}
\]

\[
\int_{-\infty}^{m} (m - y) \frac{1}{s_1} \phi \left( \frac{y - \mu}{s_1} \right) \Phi \left( \frac{m - y - a + s_2 \rho (y - \mu) / s_1}{\sqrt{(1 - \rho^2) s_2^2}} \right) \, dy
\]

\[
= (m - \mu) \int_{0}^{\Phi \left( \frac{m - \mu}{s_1} \right)} \Phi \left( \frac{m - \mu - a + (s_2 \rho - s_1) \Phi^{-1}(w)}{\sqrt{(1 - \rho^2) s_2^2}} \right) \, dw
\]

\[
+ s_1 \Phi \left( \frac{m - \mu}{s_1} \right) \phi \left( \frac{-a + s_2 \rho (m - \mu) / s_1}{\sqrt{(1 - \rho^2) s_2^2}} \right)
\]

\[
+ s_1 \sqrt{A} \frac{s_1 - s_2 \rho}{\sqrt{2\pi(1 - \rho^2) s_2^2}}
\]
\[
\times \exp \left\{ -\frac{1}{2} \left[ \frac{\mu^2}{s_1^2} + \frac{(m - a + \rho s_2 \mu/s_1)^2}{\sqrt{(1 - \rho^2)s_2^2}} \right] \right\} \\
\times \exp \left\{ \frac{1}{2} A v^2 \right\} \Phi \left( \frac{m - Av}{\sqrt{A}} \right), 
\] (B.2.4)

where \( v, r, \) and \( A \) are defined in Result 2.

Proof. First, use integration by parts with

\[
u = \Phi \left( \frac{m - y - a + s_2 \rho (y - \mu)/s_1}{\sqrt{(1 - \rho^2)s_2^2}} \right)
\]

\[
du = -\frac{(1 - s_2 \rho/s_1)}{\sqrt{(1 - \rho^2)^2 s_2^2}} \phi \left( \frac{m - y - a + s_2 \rho (y - \mu)/s_1}{\sqrt{(1 - \rho^2)s_2^2}} \right)
\]

\[
v = (m - y) \frac{1}{s_1} \phi \left( \frac{y - \mu}{s_1} \right)
\]

\[
to show that (B.2.3) is equal to
\]

\[
(m - \mu) \Phi \left( \frac{m - \mu}{s_1} \right) \Phi \left( \frac{-a + s_2 \rho (m - \mu)/s_1}{\sqrt{(1 - \rho^2)s_2^2}} \right) + s_1 \phi \left( \frac{m - \mu}{s_1} \right) \Phi \left( \frac{y - \mu}{s_1} \right)
\] (B.2.5)

\[
+ s_1 \phi \left( \frac{m - \mu}{s_1} \right) \Phi \left( \frac{-a + s_2 \rho (m - \mu)/s_1}{\sqrt{(1 - \rho^2)s_2^2}} \right)
\] (B.2.6)

\[
+ \int_{-\infty}^{m} \frac{s_1 - s_2 \rho}{\sqrt{(1 - \rho^2)s_2^2}} \phi \left( \frac{m - y - a + s_2 \rho (y - \mu)/s_1}{\sqrt{(1 - \rho^2)s_2^2}} \right) \phi \left( \frac{y - \mu}{s_1} \right) dy
\] (B.2.7)

\[
+ (m - \mu) \int_{-\infty}^{m} \frac{s_1 - s_2 \rho}{\sqrt{(1 - \rho^2)s_2^2}} \phi \left( \frac{m - y - a + s_2 \rho (y - \mu)/s_1}{\sqrt{(1 - \rho^2)s_2^2}} \right) \Phi \left( \frac{y - \mu}{s_1} \right) dy
\] (B.2.8)

Using integration by parts again with

\[
u = \Phi \left( \frac{y - \mu}{s_1} \right)
\]
\[ du = \frac{1}{s_1} \phi \left( \frac{y - \mu}{s_1} \right) \]

\[ dv = \frac{1 - s_2 \rho / s_1}{\sqrt{(1 - \rho^2)^2 s_2^2}} \phi \left( \frac{m - y - a + s_2 \rho (y - \mu) / s_1}{\sqrt{(1 - \rho^2)^2 s_2^2}} \right) \]

\[ v = -\Phi \left( \frac{m - y - a + s_2 \rho (y - \mu) / s_1}{\sqrt{(1 - \rho^2)^2 s_2^2}} \right), \]

we can now express (B.2.8) as

\[ (m - \mu) \left[ -\Phi \left( \frac{m - \mu}{s_1} \right) \Phi \left( \frac{-a + s_2 \rho (m - \mu) / s_1}{\sqrt{(1 - \rho^2)^2 s_2^2}} \right) \right. \]

\[ + \int_{-\infty}^{m} \frac{1}{s_1} \phi \left( \frac{y - \mu}{s_1} \right) \Phi \left( \frac{m - y - a + s_2 \rho (y - \mu) / s_1}{\sqrt{(1 - \rho^2)^2 s_2^2}} \right) dy. \]  \hspace{1cm} (B.2.9)

Summing (B.2.5), (B.2.6), (B.2.7), and (B.2.9), we see that (B.2.3) is equal to

\[ (m - \mu) \int_{-\infty}^{m} \frac{1}{s_1} \phi \left( \frac{y - \mu}{s_1} \right) \Phi \left( \frac{m - y - a + s_2 \rho (y - \mu) / s_1}{\sqrt{(1 - \rho^2)^2 s_2^2}} \right) dy \]  \hspace{1cm} (B.2.10)

\[ + s_1 \phi \left( \frac{m - \mu}{s - 1} \right) \Phi \left( \frac{-a + s_2 \rho (m - \mu) / s_1}{\sqrt{(1 - \rho^2)^2 s_2^2}} \right) \]  \hspace{1cm} (B.2.11)

\[ + \int_{-\infty}^{m} s_1 - s_2 \rho / \sqrt{(1 - \rho^2)^2 s_2^2} \phi \left( \frac{y - \mu}{s_1} \right) \Phi \left( \frac{m - y - a + s_2 \rho (y - \mu) / s_1}{\sqrt{(1 - \rho^2)^2 s_2^2}} \right) dy. \]  \hspace{1cm} (B.2.12)

To complete the proof, apply Result 2 to (B.2.12) and employ the change of variable

\[ w = \Phi \left( \frac{y - \mu}{s_1} \right) \] to (B.2.10). \hfill \Box

**Proof of Theorem 5.** The goal is to show \( \text{Int}_{i,j} = \text{Int}_{i,j}^1 + \text{Int}_{i,j}^2 + \text{Int}_{i,j}^3 \), from which the desired result directly follows. To do so, we simplify the 2p integrals \( \text{Int}_{i,j} \). First, realize that

\[ \text{Int}_{i,j} \]

\[ = \int_{-\infty}^{y_i(x_j^*)} \int_{y_{k(i)}(x_j^*) - y_i(x_j^*) + y_i} f(y_1, y_2) dy_k(i) dy_i 
- \int_{y_i(x_j^*)}^{y_i(x_j^*)} \int_{y_{k(i)}(x_j^*) - y_i(x_j^*) + y_i} f(y_1, y_2) dy_k(i) dy_i. \]
We can rewrite this expression using the fact that $f(u_1, u_2) = f(u_1 | u_2) f(u_2)$ for any joint pdf $f(u_1, u_2)$ and the fact that

$$U_1 | U_2 = u_2 \sim N \left( \mu_1 + \frac{\sigma_1}{\sigma_2} \rho (u_2 - \mu_2), (1 - \rho^2) \sigma_1^2 \right)$$  \hspace{1cm} (B.2.13)

if

$$\begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \sim N \left( \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \sigma_1^2 & \sigma_1 \sigma_2 \rho \\ \sigma_1 \sigma_2 \rho & \sigma_2^2 \end{bmatrix} \right).$$

We then have

$$\begin{align*}
\text{Int}_{i,j} &= \int_{y_i(x_j^*)}^{y_i(x_j^*)} (y_i(x_j^*) - y_i) \\
& \times \left[ \int_{y_k(x_j)}^{y_k(x_j)} 1 \right. \\
& \times \phi \left( \frac{y_k(x_j) - \hat{y}_k(x_j) - \rho(x) s_k(x_k) (y_i - \hat{y}_i(x)) / s_i(x)}{\sqrt{1 - \rho^2(x) s_k^2(x)}} \right) dy_k(i) \\
& \times \frac{1}{s_i(x)} \phi \left( \frac{y_i - \hat{y}_i(x)}{s_i(x)} \right) dy_i \\
& \left. - \int_{y_i(x_j^*)}^{y_i(x_j^*)} (y_i(x_j^*) - y_i) \\
& \times \left[ \int_{y_k(x_j)}^{y_k(x_j)} 1 \right. \\
& \times \phi \left( \frac{y_k(x_j) - \hat{y}_k(x_j) - \rho(x) s_k(x_k) (y_i - \hat{y}_i(x)) / s_i(x)}{\sqrt{1 - \rho^2(x) s_k^2(x)}} \right) dy_k(i) \\
& \times \frac{1}{s_i(x)} \phi \left( \frac{y_i - \hat{y}_i(x)}{s_i(x)} \right) dy_i \right]
\end{align*}$$

Now, express the inner integrals in terms of the normal cdf, and use the fact that

$1 - \Phi(z) = \Phi(-z)$ to yield

$$\text{Int}_{i,j}$$

306
\[
\begin{align*}
&= \int_{-\infty}^{y_i(x_j^*)} \left( y_i(x_j^*) - y_i \right) \\
&\times \Phi \left( \frac{y_i(x_j^*) - y_i - (y_{k(i)}(x_j^*) - \hat{y}_{k(i)}(x)) + \rho(x)s_{k(i)}(x)(y_i - \hat{y}_i(x)) / s_i(x)}{\sqrt{1 - \rho^2(x)s_{k(i)}^2(x)}} \right) \\
&\times \phi \left( \frac{y_i - \hat{y}_i(x)}{s_i(x)} \right) dy_i \\
&- \int_{-\infty}^{y_i(x_j^*)} \left( y_i(x_j^*) - y_i \right) \\
&\times \Phi \left( \frac{y_i(x_j^*) - y_i - (y_{k(i)}(x_{h(i,j)}) - \hat{y}_{k(i)}(x)) + \rho(x)s_{k(i)}(x)(y_i - \hat{y}_i(x)) / s_i(x)}{\sqrt{1 - \rho^2(x)s_{k(i)}^2(x)}} \right) \\
&\times \phi \left( \frac{y_i - \hat{y}_i(x)}{s_i(x)} \right) dy_i 
\end{align*}
\]

The final step is to apply Result 3 to (B.2.14) with \( y = y_i, a = d(k(i), j), m = y_i(x_j^*), \mu = \hat{y}_i(x), \rho = \rho(x), s_1 = s_i(x), \) and \( s_2 = s_{k(i)}(x) \) and then to (B.2.15) with \( y = y_i, a = d(k(i), h(i,j)), m = y_i(x_j^*), \mu = \hat{y}_i(x), \rho = \rho(x), s_1 = s_i(x), \) and \( s_2 = s_{k(i)}(x). \)
APPENDIX C

SOFTWARE MANUALS

C.1 NSDep User’s Guide

As a quick review of notation, recall that we are modelling an m-dimensional function \( y(\cdot) = (y_1(\cdot), \ldots, y_m(\cdot)) \) from a d-dimensional input space \( \mathcal{X} \in \mathbb{R}^d \) with positive d-dimensional volume. Our data consists an initial design \( \mathcal{D}_n = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \subset \mathcal{X} \) and the evaluations of \( y(\cdot) \) at this initial design. Let \( \mathbf{y}^{m,n} = (\mathbf{y}^T(\mathbf{x}_1), \ldots, \mathbf{y}^T(\mathbf{x}_n))^T \) be the vector of all outputs at \( \mathcal{D}_n \). We assume that \( y(\cdot) \) can be modelled as an m-variate Gaussian process \( Y(\cdot) \).

The MATLAB function \( \text{nsdep.m} \) assumes \( Y(\cdot) \) follows a nonseparable linear model of coregionalization, as described in Section 1.3.2. The function assumes the process \( Y(\cdot) \) has a constant mean for each output of the computer code (i.e., \( E[Y(\mathbf{x})] = \beta = (\beta_1, \ldots, \beta_m)^T \)), and it assumes that the stationary latent processes \( Z(\mathbf{x}) \) in (1.3.28) each have Gaussian covariance functions, as described in (1.3.30). In addition to fitting the model at an arbitrary design \( \mathcal{D}_n \) (i.e., estimating the unknown parameters \( \mathbf{A} \) and \( \mathbf{\theta} \)), \( \text{nsdep.m} \) will also obtain predictions of the function at any number of inputs, as well as estimates of their prediction error. More specifically, if predictions are desired at \( \mathbf{x}_1^0, \ldots, \mathbf{x}_{na}^0 \), then the function returns the posterior predictive means.
\( \hat{y}(x_1^0), \ldots, \hat{y}(x_{n_0}^0) \) and posterior predictive covariance matrices \( S(x_1^0), \ldots, S(x_{n_0}^0) \) as described in (1.3.8) - (1.3.12). (The predictive means and covariances will have the true parameters replaced by REML estimates.)

To call \texttt{nsdep.m}, one would use the following syntax:

```matlab
output = nsdep(D,XPred,Y,Ysf,init,PopSize,Gen)
```

Here is a description of the various inputs:

1. \( D \) is the \( n \times d \) design \( D_n \).

2. \( XPred \) is the \( n_0 \times d \) matrix of points where predictions are desired. Each row is a different point in the input space. (We denote these points as \( x_1^0, \ldots, x_{n_0}^0 \) above.) This input must be set to the empty vector \( [] \) if no predictions are wanted.

3. \( Y \) is the \( n \times m \) matrix of outputs at the design \( D_n \) (i.e., a nonvectorized version of \( y_{m,n} \)).

4. \( Ysf \) is a \( n_{sf} \times m \) matrix of outputs that are used to scale the outputs, as described in Appendix C.1.2. (Often, this will be the same as \( Y \), especially if the initial design is space-filling.) We will refer to this set of outputs as \( Y^{sf} \) in the next section.

5. \( init \) is an initial guess at the unknown scaled parameters (see Appendix C.1.2). (Usually, this will be left empty.)

6. \( PopSize \) is the size of the populations used in the genetic algorithm that is used in one stage of the REML estimation of parameters (see Appendix C.1.2).
7. Gen is the number of generations used in the genetic algorithm that is used in one stage of the REML estimation of parameters (see Appendix C.1.2).

The output of nsdep.m is a MATLAB structure, say output. Here is a description of the various components of output.

1. output.Ymn is the $nm \times 1$ vector $y^{m,n}$.

2. output.Y is the $n \times m$ matrix of outputs at the initial design (i.e., a nonvectorized version of $y^{m,n}$).

3. output.D is the $n \times d$ initial design $D_n$.

4. output.Ysf is a $n_{sf} \times m$ matrix of outputs that are used to scale the outputs, as described in Appendix C.1.2. We will refer to this quantity of $Y^{sf}$.

5. output.Xstd is the $n \times d$ scaled initial design $D_n$ (Appendix C.1.2).

6. output.BhatGLS is $\hat{\beta}$ from (1.3.10) based on scaled inputs and outputs (see Appendix C.1.2).

7. output.scriptF is $F_n$ from Section 1.3.1.

8. output.FtSigmaFinv is $(F_n^T \Sigma_{m,n}^{-1} F_n)^{-1}$ in $\hat{\beta}$ from (1.3.10) based on scaled inputs and outputs.

9. output.mu is the $m \times 1$ vector of means of each column of $Y^{sf}$. Below, we denote this vector $\mu_{sf}$.

10. output.s is an $m \times m$ diagonal matrix, where the $i^{th}$ diagonal entry is the standard deviation of the $i^{th}$ column of $Y^{sf}$.
11. **output.covpars.offdiag** is a vector of the off-diagonal elements of the scaled $A$ matrix.

12. **output.covpars.betas** is $m \times d$ matrix where the $(i,j)^{th}$ element is the scaled correlation parameter $\theta_{i,j}$, as described in Appendix C.1.2. To obtain $\theta_{i,j}$ for the original inputs, one can multiply by $r_j$ and add $\min_q \{x_{q,j}\}$ from expression (C.1.1).

13. **output.covariance.Sigmamn** is a version of $\Sigma_{m,n}$ based on the scaled covariance parameters.

14. **output.covariance.Sigmamninv** is a version of $\Sigma_{m,n}^{-1}$ based on the scaled covariance parameters.

15. **output.covariance.CorrY** is the correlation matrix corresponding to covariance matrix $AA$.

16. **output.covariance.SigmaOUT** is the estimated covariance matrix of $Y(x)$ based on scaled inputs and outputs.

17. **output.covariance.lambda** is a $nm \times nm$ MATLAB cell matrix, where the $(i,j)^{th}$ cell is $\Lambda_{i,j}$, as described in (1.3.33), based on scaled covariance parameters.

18. **output.covariance.L** is the Cholesky decomposition of the estimated scaled $A$ matrix.

19. **output.covariance.A** is the estimated matrix $A$ based on scaled inputs and outputs.
20. `output.covariance.Sigma0` is the estimated covariance matrix of $Y(x)$ based on the original, unscaled inputs and outputs.

21. `output.lik.thebest` is the value of the minimized restricted likelihood function (C.1.2) found by `fmincon`.

22. `output.lik.bestcp` is the set of estimated scaled covariance parameters found by `fmincon`.

The following components are included only if `XPred` is not empty:

1. `output.pred.ypred` is $n_0 \times m$ matrix of predictions (posterior means) at `XPred`.

2. `output.pred.ypredcov` is a $n_0 \times 1$ MATLAB cell matrix, where the $i^{th}$ cell is the posterior predictive covariance matrix $S(x_0^i)$ of the $i^{th}$ prediction point $x_0^i$.

3. `output.pred.xpred` is the $n_0 \times d$ set of points where predictions are desired.
   (We denoted these inputs as $x_0^1, \ldots, x_{n_0}^0$ above.)

4. `output.covariance.Sigma0mn` is the matrix $\Sigma_{0,n_0,n}$ from (1.3.35) based on scaled inputs and outputs.

5. `output.covariance.lambda0` is the matrix $\Lambda_{0,i,j}$ from (1.3.35) based on scaled inputs and outputs.

C.1.1 Functions Called by NSDep

The function `nsdep.m` calls the following helper functions.

1. `MakeP` constructs the $mn \times mn$ matrix $P$ which is described at the end of Appendix C.1.2, and which is used to compute the inverse of $\Sigma_{m,n}$. 

312
2. `nsdepnegresloglikUS` computes the scaled restricted likelihood function (C.1.2).

3. `MakeBDCovZ` constructs the $mn \times mn$ matrix

$$\text{blkdiag} \left( \Sigma_{n}^{Z_{1}}, \ldots, \Sigma_{n}^{Z_{m}} \right).$$

described in Appendix C.1.2.

4. `MakeL` and `Makebetas` takes the vector of scaled estimated covariance parameters, which are stored as an $dm + \frac{m(m+1)}{2}$ vector

$$\left( [\hat{L}^{*}]_{11}, \ldots, [\hat{L}^{*}]_{mm}, [\hat{L}^{*}]_{21}, [\hat{L}^{*}]_{31}, \ldots, [\hat{L}^{*}]_{m-2,m}, [\hat{L}^{*}]_{m-1,m}, \hat{\theta}^{*}_{1,1}, \ldots, \hat{\theta}^{*}_{1,d}, \ldots, \hat{\theta}^{*}_{m,1}, \ldots, \hat{\theta}^{*}_{m,d} \right)$$

when output from the optimization routines (see Appendix C.1.2), and places these elements into matrix form.

5. `MakeLambda` creates a MATLAB cell matrix where the $(i,j)^{th}$ cell contains the matrix

$$\Lambda_{i,j} = \bigoplus_{k=1}^{m} R_{k}(x_{i} - x_{j} ; \theta_{k}).$$

6. `MakeSigmaMn` constructs the estimated matrix $\Sigma_{m,n}$.

7. `MakeSigmaMnInv` calculates the estimated inverse of $\Sigma_{m,n}$.

8. `MakeBhatGLS` creates the $m \times 1$ vector $\hat{\beta}$.

### C.1.2 Some Details Regarding REML Estimation

Here we will describe briefly the strategy for obtaining REML estimates for the $dm + \frac{m(m+1)}{2}$ parameters in $A$ and $\theta$, where $m$ is the dimension of the output space, and $d$ is the dimension of the input space. To find the REML estimates, we need to solve the optimization problem described in (1.3.37).
First, the observed outputs $y^{m,n}$ are scaled as
\[(y^{m,n} - \mu_{sf}) S_{sf}^{-1}\]
where $\mu_{sf}$ is the sample mean vector of $Y^{sf}$ and $S_{sf}$ is the $m \times m$ diagonal matrix whose $i^{th}$ diagonal element is the sample standard deviation of the values corresponding to the $i^{th}$ output in $Y^{sf}$. Also, letting $x_{q\ell}$ be the $\ell^{th}$ dimension of the $q^{th}$ input in the design, the inputs are scaled as
\[
\frac{x_{q,\ell} - \min_q \{x_{q\ell}\}}{r_{\ell}}
\]
where $r_{\ell} = \max_q \{x_{q\ell}\} - \min_q \{x_{q\ell}\}$. This forces the design points to lie on the hypercube $[0,1]^d$. The algorithm estimates parameters based on the scaled inputs and scaled outputs. We denote these scaled parameters by $A^*$ and $\theta^*$. Also, as mentioned in Chapter 1, we will find estimates of $L^*$, where $L^*$ is the Cholesky decomposition of the positive definite matrix $A^*$ so that $A^* = L^* (L^*)^T$, since the only restrictions on $L^*$ is that it have positive diagonals. The maximization of the scaled restricted log-likelihood function
\[
L(L^*, \theta^*) = -\frac{1}{2} \log (|\Sigma_{m,n}|) - \frac{1}{2} \log (F_n^T \Sigma_{m,n}^{-1} F_n) - \frac{1}{2} \left(y^{m,n} - F_n \hat{\beta}_{GLS}\right)^T \Sigma_{m,n}^{-1} \left(y^{m,n} - F_n \hat{\beta}_{GLS}\right)
\]
occsurs in several stages.

The first step is to create artificial bounds on all parameters. This allows us to use some direct search optimization tools for constrained optimization to hone in on “promising” regions of the parameter space. Specifically, the lower bound for $\theta_{i,j}^*$, where $i = 1, \ldots, m$ and $j = 1, \ldots, d$, is set so that, for the latent process $Z_i(\cdot)$, the correlation between the two (hypothetical) points obtained by projecting the training inputs into each dimension is between 0.01 and 0.99.
More specifically, for the Gaussian correlation function let $M_j$ denote the largest $L_2$ Euclidean distance between two training data inputs in $j$th dimension, and $m_j$ is the smallest non-zero $L_2$ distance between two training data inputs in the $j$th dimension, $j = 1, \ldots, d$. Then we take the lower bound for each $\theta^*_{i,j}$ to be
\[-\log(0.99^{1/d})/M_j\]
and the upper bound for each $\theta^*_{i,j}$ to be
\[-\log(0.01^{1/d})/m_j,\]
where $d$ is the number of inputs. Also, the diagonals of $L^*$ are assume to lie on $[0, 4]$, and the off-diagonal elements of $L^*$ are assumed to lie on $[-4, 4]$. These bounds on the elements of $A^*$ should be reasonable, given that we are dealing with a standardized version of the original outputs.

After creating the bounds, a $20 \times \left( dm + \frac{m(m+1)}{2} \right)$ point random Latin hypercube is generated in this bounded parameter space, and $L(\cdot, \cdot)$ is evaluated at all these points. The PopSize points with the greatest restricted likelihood evaluations are retained, and the rest are discarded. These PopSize points serve as the initial population in the call to the MATLAB function `ga.m` from Forrester et al. (2008). The algorithm `ga.m` is run for Gen generations, using the same bounds that were used to create the random Latin hypercube of inputs. Finally, the MATLAB function `fmincon` is called, with the starting point set equal to the maximizer found using `ga.m`. The same bounds that were used for `ga.m` are also used for `fmincon`. The options for `fmincon` are set by the following command:

\[
\text{options = optimset('Display','off','MaxIter',50000,...}
\text{'MaxFunEvals',10000,'LargeScale','Off',...}
\text{'Algorithm','active-set','TolFun',.0001);}\]
The above describes how estimation of the unknown parameters occurs if the input init is empty. If a \((dm + \frac{m(m+1)}{2})\)-dimensional vector is specified for init, then this initial guess of the parameters is used as the starting point for fmincon, and the genetic algorithm is not used to find a starting point for fmincon.

Finally, we note that evaluations of \(L(\cdot, \cdot)\) require inversion of \(\Sigma_{m,n}\), an \(mn \times mn\) symmetric, positive definite; \(\Sigma_{m,n}\) can be expressed as

\[
\begin{pmatrix}
\Sigma_0 & Cov(Y(x_1), Y(x_2)) & \cdots & Cov(Y(x), Y(x_n)) \\
Cov(Y(x_1), Y(x_2)) & \Sigma_0 & \cdots & Cov(Y(x_2), Y(x_n)) \\
\vdots & \vdots & \ddots & \vdots \\
Cov(Y(x_1), Y(x_n)) & Cov(Y(x_2), Y(x_n)) & \cdots & \Sigma_0 \\
\end{pmatrix} = 
\begin{pmatrix}
AA & A\Lambda_{1,2} & \cdots & A\Lambda_{1,n} \\
A\Lambda_{1,2} & AA & \cdots & A\Lambda_{2,n} \\
\vdots & \vdots & \ddots & \vdots \\
A\Lambda_{1,n} & A\Lambda_{2,n} & \cdots & AA \\
\end{pmatrix} = \begin{pmatrix}
I_m & \Lambda_{1,2} & \cdots & \Lambda_{1,n} \\
\Lambda_{1,2} & I_m & \cdots & \Lambda_{2,n} \\
\vdots & \vdots & \ddots & \vdots \\
\Lambda_{1,n} & \Lambda_{2,n} & \cdots & I_m \\
\end{pmatrix} \begin{pmatrix}
I_n \otimes A \\
\end{pmatrix}
\]

where \(\Lambda_{i,j} = \oplus_{k=1}^{m} R_k(x_i - x_j; \theta_k)\). Letting \(\Sigma_{n}^{Z_i}\) denote the \(n \times n\) matrix whose \((j, k)\)th element is \(Cov(Z_i(x_j), Z_i(x_k))\), then, with an appropriately chosen \(mn \times mn\) permutation matrix \(P\), we have

\[
\Sigma_{m,n} \begin{pmatrix}
I_n \otimes A \\
\end{pmatrix} \begin{pmatrix}
P \text{blkdiag} \left(\Sigma_{n}^{Z_1}, \ldots, \Sigma_{n}^{Z_m}\right) \begin{pmatrix}
P^T \left(I_n \otimes A\right) \\
\end{pmatrix}\end{pmatrix}.
\]

Thus, we can calculate the desired inverse as

\[
\Sigma_{m,n}^{-1} = \begin{pmatrix}
I_n \otimes A^{-1} \\
\end{pmatrix} \begin{pmatrix}
P \text{blkdiag} \left(\left(\Sigma_{n}^{Z_1}\right)^{-1}, \ldots, \left(\Sigma_{n}^{Z_m}\right)^{-1}\right) \begin{pmatrix}
P^T \left(I_n \otimes A^{-1}\right) \\
\end{pmatrix}\end{pmatrix}.
\]

Therefore, instead of inverting an \(mn \times mn\) matrix, we need only invert \(m+1\) matrices of size \(n \times n\), which is much faster for larger \(m\).
C.1.3 Example

We use nsdep.m to fit a nonseparable LMC based on training data generated from the WSNL function described in Section 3.1.2. The training data inputs are a 20 point maximin Latin hypercube design $X$.

$$X = \begin{bmatrix}
4.3750 & 2.6250 \\
-4.6250 & 10.8750 \\
9.6250 & 12.3750 \\
8.1250 & 4.1250 \\
2.1250 & 8.6250 \\
1.3750 & 0.3750 \\
-0.8750 & 11.6250 \\
-3.8750 & 4.8750 \\
7.3750 & 1.1250 \\
2.8750 & 13.1250 \\
5.8750 & 13.8750 \\
-3.1250 & 1.8750 \\
-0.1250 & 6.3750 \\
-1.6250 & 14.6250 \\
5.1250 & 7.8750 \\
6.6250 & 10.1250 \\
3.6250 & 5.6250 \\
0.6250 & 3.3750 \\
-2.3750 & 9.3750 \\
8.8750 & 7.1250
\end{bmatrix} ;$$

The output at each input in the design is a $20 \times 2$ matrix $Y$. 

317
Predictions of $y(\cdot)$ are desired at the ten inputs specified by $X_{Pred}$.

To fit the model and obtain predictions, we call `nsdep.m` with

```
outputn = nsdep(X,XPred,Y,Y,[],10*(m*(m+1)/2+m*d),20);
```
(Here, m=2 and d=2.) Notice that \( Y \) is taken to be \( Y^{sf} \), no initial guess at the covariance parameters is provided, and the preliminary stage of the optimization algorithm uses a genetic algorithm with 20 generations of size \( 10 \times \left( \frac{m(m+1)}{2} + md \right) = 70 \). The output of \texttt{nsdep.m} is stored in the MATLAB structure \texttt{outputn}.

\begin{verbatim}
outputn =
    Ymn: [40x1 double]
    Y: [20x2 double]
    D: [20x2 double]
    Ysf: [20x2 double]
    Xstd: [20x2 double]
    BhatGLS: [2x1 double]
    scriptF: [40x2 double]
    FtSigmaFinv: [2x2 double]
    mu: [20x2 double]
    s: [20x2 double]
    covpars: [1x1 struct]
    covariance: [1x1 struct]
    lik: [1x1 struct]
    pred: [1x1 struct]
\end{verbatim}

The predictions at \( \texttt{XPred} \) are stored in \texttt{outputn.pred.ypred}.

\begin{verbatim}
outputn.pred.ypred =
    1.0e+003 *
    [1.1354 -0.4416
     2.3627 -0.1400
     0.2051 -0.2058
     0.0634 -0.0752
     0.1554 -0.1676
     1.6752 -0.4568
     1.2249 -0.4125
     0.1785 -0.2798
     2.0342 -0.5329
     1.1891 -0.1438
\end{verbatim}

319
The predictions compare favorably to the true values of the WSNL function at $X_{\text{Pred}}$, which we have stored in $Y_{\text{True}}$.

\[ Y_{\text{True}} = 1.0e+003 \times 
\begin{bmatrix}
1.1071 & -0.4349 \\
2.7237 & -0.1795 \\
0.2122 & -0.2070 \\
0.0380 & -0.0786 \\
0.1642 & -0.1670 \\
1.6666 & -0.4540 \\
1.2144 & -0.4101 \\
0.1836 & -0.2802 \\
2.0336 & -0.5352 \\
1.1989 & -0.1492
\end{bmatrix} \]

The estimated correlation of matrix of the WSNL function can be found in $\text{outputn.covariance.CorrY}$.

\begin{bmatrix}
1.0000 & -0.1699 \\
-0.1699 & 1.0000
\end{bmatrix}

The estimated matrix $\Sigma_0$ (the between-output covariance in the original scale) is found in $\text{outputn.covariance.SigmaOUT}$

\begin{bmatrix}
3.8115 & -0.1320 \\
-0.1320 & 0.1584
\end{bmatrix}

The predictive uncertainty at each point in $X_{\text{Pred}}$ can be found in the cell matrix $\text{outputn.pred.ypredcov}$. For example, the estimated posterior predictive covariance at the fourth prediction point (9.0099, 0.6926) is found in $\text{outputn.pred.ypredcov}[4]$. 

320
The MATLAB function `multiopt.m` implements expected improvement algorithms for multiobjective optimization, as described in Chapter 2. It implements (almost) every single possible combination of improvement function, $QI(x)$ interpretation, and dependence structure (independence or nonseparable LMC) as described in Chapter 1 and Chapter 2.

The function `multiopt.m` relies on NOMADm (see Abramson (2010)), a MATLAB implementation of a mesh adaptive direct search algorithm (see Audet and Dennis (2006)). The main optimization function `mads.m` is called by one of the following set-up files, depending on the choice of GP model and output dimension:

1. `mads_batch_EMMI2d.m`
2. `mads_batch_ind.m`
3. `mads_batch_dep.m`

Each of these functions requires that one specify the location essential problem files. Therefore, in order for `mads.m` and `multiopt.m` to work properly, the following functions must be in the current folder (as specified by the `pwd` command):

1. `expmaxminimp2dnomad.m`
2. `expmaxminimp2dnomad_Omega.m`
3. expmaxminimp2dnomad_x0.m
4. expimpind.m
5. expimpind_Omega.m
6. expimpind_x0.m
7. expimpdep.m
8. expimpdep_Omega.m
9. expimpdep_x0.m

The function multiopt.m is called via the following syntax:

```matlab
output = multiopt('funcname','wsnl','outputdim',m,...
    'inputdim',d,'x',X,'y',Y,'ranges',BD,...
    'sequentialdesign',Nstar,'correlationfamily','Gaussian',...
    'method','EMMI','model','Ind','manualinput','No');
```

Inputs are specified in pairs; the first item of each pair is a character string and the second is its value. Here is a description of the various inputs for multiopt.m:

1. **funcname** specifies the name of function being optimized. It is only necessary if one is not manually adding input. (See ‘manualinput’ below.) The function specified by funcname must take \( n \times d \) dimensional input and return \( n \times m \) dimensional output. (The number of inputs is \( n \), the input space dimension is \( d \), and \( m \) is the number of outputs.)

2. **outputdim** specifies the dimension \( m \) of the output space, i.e., the number of functions.
3. `inputdim` specifies the dimension $d$ of the input space.

4. `x` specifies the initial design $D_n$. It is an $n \times d$ matrix.

5. `y` specifies the $n \times m$ matrix of function evaluations at the initial design.

6. `ranges` specifies the $d \times 2$ matrix with the first column representing lower bounds on input space variables and the second column representing upper bounds on the input space variables.

7. `sequentialdesign` specifies the number $n^*$ of inputs that are added sequentially via expected improvement to the original design. The default is $n$, the number of inputs.

8. `constraintfunctions` specifies the name of the constraint function. It is only necessary if performing constrained multiobjective optimization.

9. `constraintdimension` specifies $k$, the number of constraints, and is automatically set to 0 if `constraintfunctions` is omitted.

10. `constraintbounds` is a $2 \times k$ matrix with first column representing lower bounds on the constraints and the second column representing upper bounds on the constraints.

11. `c` specifies the $n \times k$ matrix of constraint function evaluations at the initial design.

12. `method` specifies the improvement function and $QI(x)$ interpretation used. Here are the choices (which are all described in detail in Section 2.2 and Section 2.3):

   (a) `EMMI` - Maximin Improvement (QIM1) (default)

   323
13. \textit{nmc} and \textit{nmcu} specify the number of Monte Carlo draws used in the sample average approximations of various integrals within $QI(\mathbf{x})$. The specific role \textit{nmc} and \textit{nmcu} specify depends upon the value specified by \textit{method}.

(a) \textbf{EMMI} - In the case of two-dimensional output, both are ignored because $QI(\mathbf{x})$ has been derived analytically. Otherwise, \textit{nmc} corresponds to $S$ in the QIM1 implementation in Section 2.4, and \textit{nmcu} is not used.

(b) \textbf{KEI} - \textit{nmc} corresponds to $S$ in the QIM2 implementation in Section 2.2.2, and \textit{nmcu} is not used.
(c) **PI - nmc** corresponds to *S* in Section 2.2.1, and **nmcu** is not used.

(d) **EMMI2 - nmc** corresponds to *S* in the QIM1 implementation in Section 2.4, and **nmcu** is not used.

(e) **KEI1 - nmc** corresponds to *S* in the QIM1 implementation in Section 2.2.2, and **nmcu** is not used.

(f) **EMAX1 - nmc** corresponds to *S* in the QIM1 implementation in Section 2.2.4, and **nmcu** is not used.

(g) **ECI1 - nmc** corresponds to *S* in the QIM1 (averaged completeness indicator) implementation in Section 2.3.3, and **nmcu** corresponds to *T*.

(h) **ECI2 - nmc** corresponds to *S* in the QIM2 (averaged completeness indicator) implementation in Section 2.3.3, and **nmcu** corresponds to *T*. The number of $Z^i_x$ draws is always fixed at $V = 5000$.

(i) **EHI1 - nmc** corresponds to *S* and **nmcu** corresponds to *T* in the the QIM1 implementation in Section 2.2.3.

(j) **EHI2 - nmc** corresponds to *S* and **nmcu** corresponds to *T* in the the QIM2 implementation in Section 2.2.3.

(k) **EWHI1 - nmcu** corresponds to *T* in the QIM1 implementation in Section 2.3.2, and **nmc** is not used.

(l) **EWHI2 - nmc** corresponds to *S* and **nmcu** corresponds to *T* in the the QIM2 implementation in Section 2.3.2.

(m) **EECI1 - nmc** corresponds to *S* in the QIM1 (estimated completeness indicator) implementation in Section 2.3.3, and **nmcu** corresponds to *T*.
EECI2 - nmc corresponds to $S$ in the QIM2 (estimated completeness indicator) implementation in Section 2.3.3, and nmcu corresponds to $T$.

14. **model** specifies the type of multivariate GP model (Ind for an independence model with product Gaussian correlation structure or Dep for a nonseparable LMC with product Gaussian correlated latent $Z(\cdot)$ processes) that is fit to the data. The independence model is fit using MPErK (see Han et al. (2011)) and the dependence model is fit using NSDep (see Appendix C.1). In the constrained case, only an independence model can be used. All GP assume a constant mean. The default is Ind.

15. **manualinput** is set to No if a MATLAB function is supplied via the *funcname* input. Otherwise, it is set to Yes, and at each stage of the expected improvement algorithm, the user must supply the output associated with the input that maximizes $QI(x)$. The Yes option is used if the function $y(\cdot)$ is not a MATLAB function (e.g., it is implemented in another language or on a different computer than the one running multiopt.)

The inputs that are mandatory depend upon whether or not **manualinput** is set to Yes or No. If Yes is chosen, then $x$, $y$, outputdim, inputdim, and ranges are must be specified. Otherwise, *funcname* must also be specified. All other inputs are optional.

The output of `multiopt.m` is contained in the MATLAB structure `output`. Here is a description of the various components of `output`.

1. **FUNCname** is the name of the MATLAB function being optimized (not included if **manualinput** is set to Yes.)
2. PFX is a matrix of nondominated inputs $\mathcal{P}_{X}^{n+n^*}$ obtained after running the expected improvement algorithm. The number of columns is equal to the dimension of the input space, and the number of rows is equal to the size of the estimated Pareto set.

3. PFY is a matrix of nondominated outputs $\mathcal{P}_{Y}^{n+n^*}$ obtained after running the expected improvement algorithm. The number of columns is equal to the dimension of the output space, and the number of rows is equal to the size of the estimated Pareto front.

4. SeqDesign is an $(n + n^*) \times (1 + d + m)$ that contains all inputs (initial design and sequentially added points) and all observed outputs. Columns $2 - (d + 1)$ contain all inputs, $(d + 2) - (1 + d + m)$ contain all outputs, and the first column has an integer that describes when this input-output combination is observed. A 1 corresponds to an input in the initial design and its corresponding output. An integer $k > 1$ corresponds to the $(k - 1)^{th}$ sequentially added input and its corresponding output. Note that any constraint functions evaluations will be stored in additional columns.

5. Optim contains the $n^* \times 1$ dimensional MATLAB arrays newptset, which contains the input that maximized the expected improvement at each step, and newptsetEMI, which contains the negative of the expected improvement function at the value specified by newptset.

6. Pareto contains two $n^* \times 1$ matlab arrays X and Y. The former contains the current Pareto set at each iteration of the expected improvement algorithm, and
the latter contains the current Pareto front at each iteration of the expected improvement algorithm.

7. $t$ is the time, in seconds, that it took to run the algorithm.

8. $N$ is the total size of the design, initial plus sequentially added points, at completion of the algorithm.

9. $N_{star}$ is $n^*$, the number of sequentially added inputs.

10. $d$ is the dimension of the input space.

11. $m$ is the dimension of the output space.

12. method is the expected improvement algorithm specified by the input method.

The following components are only contained in output if an independence model is used:

1. perkjob is a $1 \times m$ struct array that contains the MPErK outputs (see Han et al. (2011)) for independent GP models fit to each output independently.

2. THETAhat is an $(n^*+1)\times(m*d+1)$ matrix that contains the estimated correlation parameters at each stage of the sequential design for each objective functions in the last $d+1$ columns. The first column denotes the stage in the expected improvement algorithm. If constraints are present, a corresponding THETAhatc matrix contains the same information, but for the $k$ constraint functions.

3. BETAhat is an $(n^*+1) \times (m+1)$ matrix that contains the estimated mean parameter at each stage of the sequential design for each objective function in the last $m+1$ columns. The first column denotes the stage in the expected
improvement algorithm. If constraints are present, a corresponding \( \text{BETAhatc} \) matrix contains the same information, but for the \( k \) constraint functions.

If a dependence model is used, then a \( 1 \times (n^* + 1) \) cell array \( \text{DepModel} \) is included in \textbf{output}. Each cell contains the output from NSDep after it has been used to fit the dependence model at each stage of the expected improvement algorithm. See Appendix C.1 for more details regarding NSDep output.

\section*{C.2.1 Functions Called by \texttt{MultiOpt}}

When a user calls \texttt{multiOpt.m}, several other MATLAB functions are called to implement the desired expected improvement algorithm. The functions that are called depend on the GP model being fit, the choice of expected improvement algorithm, and the dimension \( m \) of the objective space.

In all cases, dependent multivariate Gaussian process models are fit using \texttt{nsdep.m} (see C.1) and its associated helper function, and independent Gaussian process models are fit using \texttt{mperk.m} (see Han et al. (2011)) and its associated helper functions.

\textbf{1) Maximin Improvement, QIM1, 2-dimensional output}

This case is handled differently than the others, since a nearly analytic form for \( QI(x) \) exist in both the dependent and independent case. Within \texttt{multiOpt.m}, the function \texttt{mads\_batch\_EMMI2d.m} calls the main optimization function \texttt{mads.m}, and supplies it with the following helper functions:

1. \texttt{expmaxminimp2dnomad.m}

2. \texttt{expmaxminimp2dnomad\_Omega.m}

3. \texttt{expmaxminimp2dnomad\_x0.m}
expmaxminimp2dnomadOmega.m provides mads.m with the bounds on the input space. expmaxminimp2dnomad_x0.m provides mads.m with the initial points from which to start the direct search optimization algorithm. expmaxminimp2dnomad.m is the function that mads.m attempts to optimize. Its input is \( x \), a \( d \times 1 \) vector in the input space. Within this function, the mean and covariance of the distribution of \( Y(x)|Y^{m,n} = y^{m,n} \) is calculated, and the probability of satisfying the constraints (if they are indeed present) is also calculated. The output of \( \text{expmaxminimp2dnomad.m} \) is defined as the negative (since mads.m attempts to minimize, and we want to maximize) of the product of the probability of satisfying any constraints and the output of expmaxminimp2d.m. The function expmaxminimp2d.m take the current Pareto front and the parameters of the distribution of \( Y(x)|Y^{m,n} = y^{m,n} \) as input, and returns the expected maximin improvement as derived in (2.3.5). expmaxminimp2d.m actually calculates the integral (2.3.5) via the function CalcInt.m, which calculates the various additive components of the integral individually via the following functions:

1. CalcPart1.m
2. CalcPart2.m
3. CalcPart3.m
4. Part3f.m

In this case, no Monte Carlo methods are necessary.

2) All Other Cases with Independent GP Models

Within multiopt.m, the function mads_batch_ind.m calls the main optimization function mads.m, and supplies it with the following helper functions:
1. expimpind.m

2. expimpind_Omega.m

3. expimpind_x0.m.

The function expimpind_Omega.m provides mads.m with the bounds on the input space, and the function expimpind_x0.m provides mads.m with the initial points from which to start the direct search optimization algorithm. expimpind.m is the function that mads.m attempts to optimize. Its input is x, a \( d \times 1 \) vector in the input space. Within this function, the mean and covariance of the distribution of \( Y(x)|Y^{m,n} = y^{m,n} \) is calculated, and the probability of satisfying the constraints (if they are indeed present) is also calculated. The output of of expimpind.m is defined as the negative (since mads.m attempts to minimize, and we want to maximize) of the product of the probability of satisfying any constraints and a sample average approximation of the function \( QI(x) \).

3) All Other Cases with Dependent GP Models

This case is identical to the independence case, except now the helper functions are

1. expimpdep.m

2. expimpdep_Omega.m

3. expimpdep_x0.m.

Also, multiopt.m only allows one to perform \textit{unconstrained} multiobjective optimization with a dependence model.
C.2.2 Some notes on the optimization of $QI(x)$

During each iteration of the expected improvement algorithm implemented by `multiOpt.m`, $QI(x)$ (or an approximation to $QI(x)$) is maximized by `mads.m` in the NOMADm package (see Abramson (2010)). The function `mads.m` implements a mesh adaptive direct search (MADS) algorithm. Here, we will briefly review the basics of MADS algorithms and its implementation within `multiOpt.m`. For more detail, we refer the reader to Audet and Dennis (2006) and the user’s guide that accompanies Abramson (2010).

All algorithms classified as MADS algorithms have the same structure. They start with an evaluation of the function of interest at a set of initial points, and then each iteration proceeds with an optional global search step and a local poll step. In both steps, the function is evaluated over a carefully constructed mesh, with the goal of finding a point on the mesh that yields a lower function value than that of the current minimum function evaluation.

The optional search step allows for any finite set of inputs on the current mesh to be evaluated. This allows for great flexibility in the kind of search that is used, which includes, but is not limited to, Latin hypercube searches, genetic algorithms, and particle swarm optimization routines.

If the search step is unsuccessful in finding an input with a lower value of the function of interest during an iteration of the algorithm, or if the user chooses not to use the search step, then the algorithm moves on to the mandatory poll step. In this step, the function is evaluated at all mesh points adjacent to the current function evaluations. If an improved input on the mesh is found in either the search or poll step, then the mesh is coarsened for the next iteration, encouraging a more
global search. Otherwise, the mesh is refined for the next iteration, as the algorithm attempts to converge to the optimal input.

The implementation used in multiopt.m to maximize \( QI(x) \) (actually, minimize \(-QI(x)\), since the NOMADm has been implemented to minimize functions) uses both the optional search step for several iterations in addition to mandatory poll step. During the first iteration of the algorithm, a 20 point random Latin hypercube is generate over the input space as part of the search step. Also, during the first 10 iterations of the algorithm, a particle swarm optimization algorithm (see Kennedy and Eberhart (1995)) is also used during the search step. After the tenth iteration, only the local poll step is invoked. Because they appear to yield satisfactory results, the default options for the poll step, the type of mesh, the mesh coarsening and refinement factors, the termination criteria, and other NOMADm options were left unchanged.

C.2.3 Example

We use multiopt.m to approximate the Pareto front and set for the WSNL function described in Section 3.1.2.

First, we run the following code to specify the objective space dimension \( m \), the input space dimension \( d \), the size of the initial design \( N \), and the number of sequentially added points \( N_{star} \), the bound on the input space \( BD \), an initial Latin hypercube design \( X \), and the initial set of function evaluations \( Y \).
\[ m = 2; \]
\[ d = 2; \]
\[ N = 20; \]
\[ Nstar = 10; \]
\[ BD = [-5 10; 0 15]; \]
\[ XZ = \text{lhsdesign}(10*d,d,'iterations',20000,...
  'criterion','maximin','smooth','off'); \]
\[ X = XZ.*\text{repmat}((BD(:,2)'-BD(:,1)'),[size(XZ,1),1])...
  + \text{repmat}(BD(:,1'),[size(XZ,1),1]); \]
\[ Y = \text{wsnl}(X); \]

Here is the initial design \( X \):

\[
X = \\
\begin{bmatrix}
6.6250 & 8.6250 \\
9.6250 & 13.1250 \\
0.6250 & 10.1250 \\
8.8750 & 10.8750 \\
1.3750 & 14.6250 \\
3.6250 & 11.6250 \\
-3.8750 & 9.3750 \\
4.3750 & 1.8750 \\
-0.8750 & 7.8750 \\
8.1250 & 2.6250 \\
-2.3750 & 0.3750 \\
5.8750 & 12.3750 \\
2.8750 & 4.1250 \\
-4.6250 & 1.1250 \\
-1.6250 & 13.8750 \\
-3.1250 & 3.3750 \\
-0.1250 & 5.6250 \\
5.1250 & 6.3750 \\
7.3750 & 4.8750 \\
2.1250 & 7.1250 \\
\end{bmatrix};
\]

Here are the function evaluations \( Y \) at the initial design:
Now, we can call `multiopt.m`. We will use an independence model with Gaussian correlation, and will use the QIM1 implementation of the maximin improvement to generate sequentially added inputs.

```matlab
outputWSNL = multiopt('funcname','wsnl','outputdim',m,'inputdim',d,'x',X,'y',Y,'ranges', BD,'sequentialdesign', Nstar,... 'correlationfamily','Gaussian','method','EMMI','model','Ind');
```

When running the expected improvement algorithm, `multiopt.m` outputs the current stage of the algorithm, and notifies the user when it is estimated the GP parameters or attempting to maximin $QI(x)$. Now, we can call `multiopt.m`. We will use an independence model with Gaussian correlation, and will use the QIM1 implementation
of the maximin improvement to generate sequentially added inputs. Also, a timer lets the user know how long it took to run the algorithm.

Parameter Estimation, N=20
Maximizing Expected Improvement, N=20
Parameter Estimation, N=21
Maximizing Expected Improvement, N=21
Parameter Estimation, N=22
Maximizing Expected Improvement, N=22
Parameter Estimation, N=23
Maximizing Expected Improvement, N=23
Parameter Estimation, N=24
Maximizing Expected Improvement, N=24
Parameter Estimation, N=25
Maximizing Expected Improvement, N=25
Parameter Estimation, N=26
Maximizing Expected Improvement, N=26
Parameter Estimation, N=27
Maximizing Expected Improvement, N=27
Parameter Estimation, N=28
Maximizing Expected Improvement, N=28
Parameter Estimation, N=29
Maximizing Expected Improvement, N=29
Parameter Estimation, N=30
Elapsed time is 550.830873 seconds.

The output from multiopt.m is stored in outputWSNL.
The estimated Pareto set is stored in the matrix `PFX`.

```
outputWSNL =

    FUNCname: 'wsnl'
    PFX: [12x2 double]
    PFY: [12x2 double]
    SeqDesign: [30x5 double]
    THETAhat: [11x5 double]
    BETAhata: [11x3 double]
    perkjob: [1x2 struct]
    Pareto: [1x11 struct]
        Optim: [1x10 struct]
            t: 543.6460
    CorrFamily: 'Gaussian'
            k: 12
            N: 30
            Nstar: 10
            d: 2
            m: 2
    method: 'EMMI'
```

The estimated Pareto front is stored in the matrix `PFY`.

```
outputWSNL.PFX

ans =

    [1.3750   14.6250
 -1.6250   13.8750
 -0.8477   14.3938
  1.2244   13.6293
 -2.8543   11.1793
  4.1148   15.0000
 -2.2581   13.7898
  3.3458   15.0000
 -1.5183   15.0000
  2.6168   15.0000
 -0.5930   15.0000
 -2.9586   13.4188];
```

The estimated Pareto front is stored in the matrix `PFY`.  

337
outputWSNL.PFY

ans =

  1.0e+003 *

   1.4097   -0.5289
   0.3874   -0.4107
   0.7379   -0.4573
   1.1559   -0.5008
   0.0110   -0.3081
   2.0891   -0.5596
   0.1873   -0.3765
   1.8990   -0.5584
   0.5705   -0.4377
   1.7396   -0.5539
   0.9370   -0.4814
   0.0348   -0.3320;


