ABSTRACT

HOSKING, MICHAEL ROBERT. Kriging for Simulation Metamodeling: Experimental Design, Reduced Rank Kriging, and Omni-Rank Kriging. (Under the direction of Dr. Stephen Roberts).

This dissertation improves an analyst’s use of simulation by offering improvements in the utilization of kriging metamodels. There are three main contributions. First an analysis is performed of what comprises good experimental designs for practical (non-toy) problems when using a kriging metamodel. Second is an explanation and demonstration of how reduced rank decompositions can improve the performance of kriging, now referred to as reduced rank kriging. Third is the development of an extension of reduced rank kriging which solves an open question regarding the usage of reduced rank kriging in practice. This extension is called omni-rank kriging. Finally these results are demonstrated on two case studies.

The first contribution focuses on experimental design. Sequential designs are generally known to be more efficient than “one shot” designs. However, sequential designs require some sort of pilot design from which the sequential stage can be based. We seek to find good initial designs for these pilot studies, as well as designs which will be effective if there is no following sequential stage. We test a wide variety of designs over a small set of test-bed problems. Our findings indicate that analysts should take advantage of any prior information they have about their problem’s shape and/or their goals in metamodeling. In the event of a total lack of information we find that Latin hypercube designs are robust default choices. Our work is most distinguished by its attention to the higher levels of dimensionality.
The second contribution introduces and explains an alternative method for kriging when there is noise in the data, which we call reduced rank kriging. Reduced rank kriging is based on using a reduced rank decomposition which artificially smoothes the kriging weights similar to a nugget effect. Our primary focus will be showing how the reduced rank decomposition propagates through kriging empirically. In addition, we show further evidence for our explanation through tests of reduced rank kriging’s performance over different situations. In total, reduced rank kriging is a useful tool for simulation metamodeling.

For the third contribution we will answer the question of how to find the best rank for reduced rank kriging. We do this by creating an alternative method which does not need to search for a particular rank. Instead it uses all potential ranks; we call this approach omni-rank kriging. This modification realizes the potential gains from reduced rank kriging and provides a workable methodology for simulation metamodeling.

Finally, we will demonstrate the use and value of these developments on two case studies, a clinic operation problem and a location problem. These cases will validate the value of this research. Simulation metamodeling always attempts to extract maximum information from limited data. Each one of these contributions will allow analysts to make better use of their constrained computational budgets.
Kriging for Simulation Metamodeling: Experimental Design, Reduced Rank Kriging, and Omni-Rank Kriging

by

Michael Robert Hosking

A dissertation submitted to the Graduate Faculty of North Carolina State University in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Operations Research

Raleigh, North Carolina

2015

APPROVED BY:

Dr. Stephen Roberts
Committee Chair

Dr. Javad Taheri

Dr. Brian Reich

Dr. James Wilson
BIOGRAPHY

Michael Hosking grew up in Chapel Hill, NC. He worked at the Chapel Hill-Carrboro YMCA as a counselor, youth sport referee, and assistant camp director. The Y remains an important part of his life.

After attending public school for K-12 in Chapel Hill, he went UNC-Chapel Hill for college. After completing his BS and MS at UNC-CH, he went onto study at North Carolina State University, receiving his PhD. There he got to work with a variety of talented individuals who helped him learn about the practice of operations research.

After completion of this dissertation, he accepted a position with Netjets and moved to Columbus, OH. He enjoys playing sand volleyball, volunteering at the Y, and watching sports.
ACKNOWLEDGMENTS

There are many people I who I would like to thank for their help in the creation of this dissertation; and more importantly for their contributions to my education and development as a person. I haven't thanked you all enough; please allow me to begin to ameliorate that here. Sadly this list is not exhaustive, thank you anyone I've forgotten.

First, thank you Dr. Roberts. Your keen questions and suggestions have made this product substantially better and taught me more about what I believe as a mathematician than most of the courses I have taken. But more than for your substantial contributions to the ideas here, I would like to thank you for your support and encouragement throughout my time at NCSU. I am forever grateful that you worked with me on this project. Also, thank you to the other members of my committee. You time and service is greatly appreciated. Your concerns and outside perspectives have made this dissertation much more robust and fruitful. Thank you for your time and effort.
Creating this has been quite a marathon. I have needed to utilize parts of my schooling that I assumed were dead and buried to complete this. I would like to thank some of the educators who taught me math I will never forget and lessons that shaped me as a person more than the material in class ever could.

- At NCSU, Drs. Denton, Fang, Hodgson, Ivy, and King: thank you all for some of the most interesting and enjoyable classes I’ve had. Your classes showed the application of advanced mathematics in engaging and enlightening ways.

- At UNC: Drs. Argon, Kulkarni, Tolle, and Ziya: you taught the classes which introduced me to operations research. Through your classes and teaching I found what I want to do for the rest of my life. I am forever in your debt, thank you.

- Before: Cynthia Ammons, Keith Cooper, and Burton “Bud” Stuart: Ms. Ammons and Mr. Stuart you were two of the most enjoyable and captivating teachers of math I’ve ever had at any level of school. The way you would cast complex ideas into concrete examples shaped how I have viewed math ever since. Mr. Cooper, your class was probably the single most important class I have ever taken. It was where I moved from simply liking math because I was good at it to loving math because of the power of what it can accomplish and beauty inherent to mathematics itself. Thank you.
Drs. Foster and Uzsoy, thank you. I never had the pleasure of your classes, but this would never have happened without you. You both guided me on how to put thoughts to page and publish results for a wider audience. You gave me work which helped me grow as a thinker. You are two of the most helpful people I have ever had the pleasure to know.

Thank you to the staff, who have always been far more patient, helpful, and friendly than I deserve: Bill, Cecilia, and most especially Linda. Linda Smith, I saw you more times than anyone else in my time here at NCSU. You were a constant and reliable source of knowledge who saved me from more problems than I care to remember, thank you.

My fellow students, thank you so much. You all were a valuable wellspring of new perspectives and ideas which pushed me past some of the worst obstacles I ever encountered. My only regret with respect to you all is not getting to know you sooner.

My friends, to be frank, without your interventions this may well have been done years ago. In fairness though, without your interventions I probably would have gone (fully) crazy and never finished. Working with many of you at the Y kept me grounded and fulfilled me in a way that prevented me from burning out on my academics. Thank for your friendship and support. I treasure you all.
Thank you to my grandmother, Georgia. Your examples of kindness, faith, and love have been an inspiration to me. Your unconditional support and positivity is something I am completely grateful for. I hope I can be as helpful to those who come after me as you were to those who came after you.

Thank you to my father, Jim. Your example of reasoned analysis is at the core of my success. I wish you had gotten to read this. You would have had great ideas on how to make it better. Better, that is what I thank you most for. That is the thing which I have decided will be my life's pursuit, the science of better. From you I received the desire to never settle or be content; the desire to be good, better, best. Thank you for giving me goals to strive for and a fire to achieve.

Thank you to my mother, Jerry. It would take the next couple hundred pages to fully express my gratitude towards you, so allow me to summarize in this space. You are the strongest and most resilient person I know. You have given me my values and my character. I don't know who I would be without you and I'm glad I don't have to find out. You are the person who has had the greatest impact on my life. Thank you for all the ways you have helped, both those I understood and those I didn't. Your support of me through my entire education has been a gift for which I will never be able to repay to you, thank you.
TABLE OF CONTENTS

LIST OF TABLES ................................................................................................................................................. x
LISTS OF SYMBOLS AND ABBREVIATIONS .............................................................................................................. xv
CHAPTER 1: INTRODUCTION ........................................................................................................................................ 1
  1.1 Dissertation Statement ....................................................................................................................................... 1
  1.2 Motivation .......................................................................................................................................................... 1
  1.3 Literature Review ............................................................................................................................................... 2
    1.3.1 Literature Review on Simulations .................................................................................................................. 3
    1.3.2 Literature Review on Metamodeling ............................................................................................................. 7
    1.3.3 Literature Review on Spatial Statistics and Kriging ....................................................................................... 14
    1.3.4 Stochastic Kriging ....................................................................................................................................... 23
  1.4 Planned Contributions to Literature .................................................................................................................... 26
CHAPTER 2: EXPERIMENTAL DESIGN FOR SIMULATION METAMODELING USING KRIGING ................................................................. 30
  2.1 Abstract .............................................................................................................................................................. 30
  2.2 Motivation .......................................................................................................................................................... 30
  2.3 Literature Review on Experimental Design for Kriging ....................................................................................... 32
    2.3.1 Experimental Design in Statistics for Kriging ............................................................................................... 32
    2.3.2 Experimental Design for SM in OR .............................................................................................................. 34
  2.4 Methods ............................................................................................................................................................ 37
    2.4.1 Choosing Designs ....................................................................................................................................... 38
    2.4.2 Choosing Testbed Problems ......................................................................................................................... 45
    2.4.3 Metamodeling Construction ......................................................................................................................... 48
    2.4.4 Metamodel Evaluation .................................................................................................................................. 50
    2.4.5 Experimental Procedure ................................................................................................................................ 56
  2.5 Results .............................................................................................................................................................. 61
    2.5.1 Problem 1: Slope ......................................................................................................................................... 63
    2.5.2 Problem 2: Hyper-J ..................................................................................................................................... 64
    2.5.3 Problem 3: Newsstand ................................................................................................................................. 66
    2.5.4 Problem 4: Queueing Network ..................................................................................................................... 67
    2.5.5 Summary of Results by Problem ................................................................................................................ 68
### CHAPTER 3: REDUCED RANK KRIGING: DEMONSTRAION AND EXPLANATION

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Abstract</td>
<td>77</td>
</tr>
<tr>
<td>3.2</td>
<td>Motivation</td>
<td>77</td>
</tr>
<tr>
<td>3.2.1</td>
<td>A Thought Experiment</td>
<td>78</td>
</tr>
<tr>
<td>3.3</td>
<td>Literature Review on Numerical Methods for Kriging</td>
<td>91</td>
</tr>
<tr>
<td>3.4</td>
<td>Analytical Modifications for RRK</td>
<td>94</td>
</tr>
<tr>
<td>3.5</td>
<td>Methods</td>
<td>100</td>
</tr>
<tr>
<td>3.6</td>
<td>How RRK Propagates though Kriging</td>
<td>102</td>
</tr>
<tr>
<td>3.7</td>
<td>Results</td>
<td>119</td>
</tr>
<tr>
<td>3.7.1</td>
<td>RRK Residuals as a Function of Rank</td>
<td>121</td>
</tr>
<tr>
<td>3.7.2</td>
<td>Various Norms for the Residuals of RRK Predictions</td>
<td>127</td>
</tr>
<tr>
<td>3.7.3</td>
<td>Effects of Dimensionality, Noise, and Number of Data Points on RRK</td>
<td>135</td>
</tr>
<tr>
<td>3.8</td>
<td>Conclusions and Extensions</td>
<td>139</td>
</tr>
</tbody>
</table>

### CHAPTER 4: OMNI-RANK KRIGING

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Abstract</td>
<td>142</td>
</tr>
<tr>
<td>4.2</td>
<td>Motivation and Introduction</td>
<td>142</td>
</tr>
<tr>
<td>4.3</td>
<td>Literature Review and Previous Work</td>
<td>143</td>
</tr>
<tr>
<td>4.4</td>
<td>Cross Validation</td>
<td>144</td>
</tr>
<tr>
<td>4.5</td>
<td>Omni-Rank Kriging</td>
<td>152</td>
</tr>
<tr>
<td>4.6</td>
<td>Omni-Rank Kriging versus Stochastic Kriging</td>
<td>156</td>
</tr>
<tr>
<td>4.6.1</td>
<td>The Base Problem</td>
<td>158</td>
</tr>
<tr>
<td>4.6.2</td>
<td>Factors Which Affect Metamodel Superiority</td>
<td>164</td>
</tr>
<tr>
<td>4.6.3</td>
<td>Effect of Dimensionality of the Problem</td>
<td>166</td>
</tr>
<tr>
<td>4.6.4</td>
<td>Effect of Noisiness of the Data</td>
<td>168</td>
</tr>
<tr>
<td>4.7</td>
<td>Conclusions and Extensions</td>
<td>169</td>
</tr>
</tbody>
</table>

### CHAPTER 5: CASE STUDIES

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>171</td>
</tr>
<tr>
<td>5.2</td>
<td>PRIME Clinic</td>
<td>172</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Clinic Problem Specifics</td>
<td>172</td>
</tr>
</tbody>
</table>
5.2.3 Clinic Problem Results, Residuals of Prediction Norms ........................................ 177
5.2.4 Clinic Problem Results, Alpha Criteria ................................................................. 180
5.2.5 Clinic Problem Results, Efficient Frontiers .......................................................... 181
5.2.6 Clinic Problem Results, Agreement with Earlier Results ...................................... 184
5.3 Ambulance Location ..................................................................................................... 184
  5.3.1 Ambulance Location Problem Specifics ................................................................. 185
  5.3.2 Testing Methodology ............................................................................................... 186
  5.3.3 Case Study Results ................................................................................................. 188
CHAPTER 6: CONCLUSIONS .............................................................................................. 194
REFERENCES ..................................................................................................................... 198
APPENDICES ....................................................................................................................... 208
APPENDIX A: BRIEF LITERATURE REVIEW ON EXPERIMENTAL DESIGN .............. 209
APPENDIX B: ILLUSTRATIONS OF THE DESIGNS FROM CHAPTER 2 .................... 216
APPENDIX C: TESTBED PROBLEM FORMULATIONS AND ILLUSTRATIONS ...... 241
  C.1 Slope .......................................................................................................................... 241
  C.2 Hyper-J ..................................................................................................................... 243
  C.3 Newsstand ............................................................................................................... 245
  C.4 Queueing Network .................................................................................................... 248
LIST OF TABLES

Table 1: Model Notation ........................................................................................................ 7
Table 2: Experimental Designs Summary .................................................................................. 43
Table 3: Effects of Dimensionality and Noise on FFDE’s Dominance for the Slope Problem 64
Table 4: Average Ranks for Each Metric on Hyper-J Problem .............................................. 65
Table 5: Dominance by Problem ............................................................................................. 68
Table 6: Weak Dominance by Dimensionality and Noise ......................................................... 69
Table 7: Paired Design Performance on ME and MAE ............................................................ 72
Table 8: Paired Design Performance on AC ........................................................................... 73
Table 9: Percentage Decrease in Norms (ORK from SK) ......................................................... 160
Table 10: Percentage Decrease in Norms over Many Replications (ORK from SK) .......... 161
Table 11: Percentage Decrease in Norms and Data requirements (ORK from SK) .......... 164
Table 12: Replications Required for Brute Force Simulation to Be Equivalent to Metamodel ................................................................................................................. 179
**LIST OF FIGURES**

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Modeling Hierarchy</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Relationship between Models</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>Metamodelling Decision Visualization</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>Possible Surrogates for a Function</td>
<td>52</td>
</tr>
<tr>
<td>5</td>
<td>Alpha Criterion Example</td>
<td>54</td>
</tr>
<tr>
<td>6</td>
<td>Relationship between Dimension and Distance</td>
<td>58</td>
</tr>
<tr>
<td>7</td>
<td>One Dimensional Function and Noisy Realizations</td>
<td>79</td>
</tr>
<tr>
<td>8</td>
<td>Moving Window Regression Metamodel, Window Size = 1</td>
<td>81</td>
</tr>
<tr>
<td>9</td>
<td>Moving Window Regression Metamodel, Window Size = 2</td>
<td>82</td>
</tr>
<tr>
<td>10</td>
<td>Moving Window Regression Metamodel, Window Size = 3</td>
<td>83</td>
</tr>
<tr>
<td>11</td>
<td>Moving Window Regression Metamodel, Window Size = 4</td>
<td>84</td>
</tr>
<tr>
<td>12</td>
<td>Moving Window Regression Metamodel, Window Size = 5</td>
<td>85</td>
</tr>
<tr>
<td>13</td>
<td>Moving Window Regression Metamodel, Window Size = 9</td>
<td>86</td>
</tr>
<tr>
<td>14</td>
<td>Moving Window Regression Metamodel, Window Size = 26</td>
<td>87</td>
</tr>
<tr>
<td>15</td>
<td>Quality of Metamodels over Different Moving Window Sizes</td>
<td>88</td>
</tr>
<tr>
<td>16</td>
<td>Residuals (1 Norm) over Window Size, Many Realizations of Noise</td>
<td>89</td>
</tr>
<tr>
<td>17</td>
<td>Zoomed in Subsection of Figure 16</td>
<td>90</td>
</tr>
<tr>
<td>18</td>
<td>Modified Kriging Flow, Step One</td>
<td>96</td>
</tr>
<tr>
<td>19</td>
<td>Modified Kriging Flow, Step Two</td>
<td>103</td>
</tr>
<tr>
<td>20</td>
<td>Standard Kriging Weights</td>
<td>104</td>
</tr>
<tr>
<td>21</td>
<td>Standard Kriging Weights on a Log Scale</td>
<td>105</td>
</tr>
<tr>
<td>22</td>
<td>Full Rank Kriging Weights, Many Replications on a Log Scale</td>
<td>106</td>
</tr>
<tr>
<td>23</td>
<td>High (Not Full) Rank Kriging Weights</td>
<td>107</td>
</tr>
<tr>
<td>24</td>
<td>Medium Rank Kriging Weights</td>
<td>108</td>
</tr>
<tr>
<td>25</td>
<td>Low Rank Kriging Weights</td>
<td>109</td>
</tr>
<tr>
<td>26</td>
<td>Kriging Weight Histogram Comparison</td>
<td>110</td>
</tr>
<tr>
<td>27</td>
<td>Modified Kriging Flow, Step Three</td>
<td>112</td>
</tr>
<tr>
<td>28</td>
<td>Kriging Weight Distribution, Full Rank</td>
<td>114</td>
</tr>
<tr>
<td>29</td>
<td>Kriging Weight Distribution, High Rank</td>
<td>115</td>
</tr>
<tr>
<td>30</td>
<td>Kriging Weight Distribution, Medium Rank</td>
<td>116</td>
</tr>
<tr>
<td>31</td>
<td>Kriging Weight Distribution, Low Rank</td>
<td>117</td>
</tr>
<tr>
<td>32</td>
<td>Modified Kriging Flow, Step Four</td>
<td>120</td>
</tr>
<tr>
<td>33</td>
<td>Residuals as a Function of RRK Rank</td>
<td>121</td>
</tr>
<tr>
<td>34</td>
<td>Absolute Value of Residuals as a Function of RRK Rank</td>
<td>122</td>
</tr>
<tr>
<td>35</td>
<td>Distribution of Absolute Residuals for Various Ranks in RRK</td>
<td>124</td>
</tr>
<tr>
<td>36</td>
<td>Relative CDFs of Distribution of Absolute Residuals</td>
<td>125</td>
</tr>
<tr>
<td>37</td>
<td>Residual Histogram Scaled for Importance</td>
<td>126</td>
</tr>
<tr>
<td>38</td>
<td>1 Norms for Various Ranks of RRK and Baseline Methods</td>
<td>129</td>
</tr>
<tr>
<td>39</td>
<td>Infinity Norms for Various Ranks of RRK and Baseline Methods</td>
<td>130</td>
</tr>
<tr>
<td>40</td>
<td>2 Norms for Various Ranks of RRK and Baseline Methods</td>
<td>131</td>
</tr>
</tbody>
</table>
Figure 41: 1 Norm for Various Kriging Methods, Quadratic Problem .............................................. 132
Figure 42: 2 Norm for Various Kriging Methods, Quadratic Problem .............................................. 133
Figure 43: Infinity Norm for Various Kriging Methods, Quadratic Problem ...................................... 134
Figure 44: Norms for 7-Dim Problem with 100 Data Points ................................................................. 135
Figure 45: Norms for 7-Dim Problem with 400 Data Points ................................................................. 136
Figure 46: Norms for 2-Dim Problem with 100 Data Points ................................................................. 137
Figure 47: Norms for Different Methods without Noise Free Data ......................................................... 138
Figure 48: CV Estimation of RRK’s Performance and RRK’s True Performance ................................. 146
Figure 49: CV of RRK’s Performance vs RRK’s True Performance ...................................................... 147
Figure 50: Zoom-in of Previous Figure ................................................................................................. 148
Figure 51: Fixed Rank RRK versus CV Performance .............................................................................. 149
Figure 52: Zoom-in of Previous Figure ................................................................................................. 150
Figure 53: CV’s Performance Compared to OFR’s Performance ............................................................ 151
Figure 54: RRK Prediction Trajectories (Figure 33 Reproduction) ....................................................... 153
Figure 55: Performance of CV Compared to ORK’s (1, 2, and Infinity Norms) ................................. 154
Figure 56: Performance of ORK Compared to OFR’s Performance (1, 2, and Infinity Norms) ................. 155
Figure 57: ORK’s Performance Compared to SK’s Performance ............................................................. 157
Figure 58: ORK vs SK Performance for an Instance of the Base Problem ............................................. 159
Figure 59: ORK vs SK Over Many Replications of the Base Problem .................................................. 161
Figure 60: Fitted ORK vs SK Over Many Replications of the Base Problem ........................................ 163
Figure 61: Percentage of Data ORK Requires to Make Equivalent Metamodels to SK by Dimensionality of the Problem .................................................................................................. 167
Figure 62: Percentage of Data ORK Requires to Make Equivalent Metamodels to SK by Noisiness of Data .................................................................................................................................. 168
Figure 63: Patient Flow Through the Clinic ............................................................................................ 173
Figure 64: Metamodeling Process for Each Design ............................................................................... 175
Figure 65: Metamodel Prediction Residuals ............................................................................................ 178
Figure 66: Design Effects on Metamodel Alpha Criteria ........................................................................ 181
Figure 67: Metamodel Efficient Frontiers Compared to Data from Full Simulation ............................... 182
Figure 68: Subset of Metamodel Efficient Frontiers vs Data from Full Simulation ............................... 183
Figure 69: Performance of Optimizer Methods, 3 Ambulance (6 Dim) Problem ..................................... 189
Figure 70: Zoom-in of Figure 69 ............................................................................................................. 190
Figure 71: Performance of Optimizer Methods, 5 Ambulance (10 Dim) Problem .................................... 191
Figure 72: Performance of Optimizer Methods, 8 Ambulance (16 Dim) Problem ................................. 192
Figure 73: Simple Random Sampling, 20 Points, 2 Dimensions ............................................................ 216
Figure 74: Simple Random Sampling, 80 Points, 2 Dimensions ............................................................. 217
Figure 75: Simple Random Sampling, 80 Points, 3 Dimensions ............................................................. 218
Figure 76: Factorial Design, 80 Points, 2 Dimensions ............................................................................ 219
Figure 77: Factorial Design, 64 Points, 3 Dimensions .......................................................................... 220
Figure 78: Factorial Design, 6 Points, 2 Dimensions ............................................................................. 221
Figure 79: Centered Factorial Design, 80 Points, 2 Dimensions ............................................................ 222
Figure 80: Central Composite Design, 23 Points, 2 Dimensions .......................................................... 223
Figure 81: Central Composite Design, 15 Points, 3 Dimensions ........................................ 224
Figure 82: Central Composite Design, 23 Points, 2 Dimensions ........................................ 225
Figure 83: Number Theoretic Design, 25 Points, 2 Dimensions ........................................ 226
Figure 84: Number Theoretic Design, 80 Points, 2 Dimensions ........................................ 227
Figure 85: Number Theoretic Design, 80 Points, 3 Dimensions ........................................ 228
Figure 86: Number Theoretic Design, 80 Points, 3 Dimensions ........................................ 229
Figure 87: Coffee House Design, 20 Points, 2 Dimensions ............................................... 230
Figure 88: Coffee House Design, 20 Points, 2 Dimensions ............................................... 231
Figure 89: Coffee House Design, 64 Points, 3 Dimensions ............................................... 232
Figure 90: Random Latin Hypercube Design, 20 Points, 2 Dimensions ............................ 233
Figure 91: Random Latin Hypercube Design, 20 Points, 2 Dimensions ............................ 234
Figure 92: Random Latin Hypercube Design, 80 Points, 3 Dimensions ............................ 235
Figure 93: Maxmin Latin Hypercube Design, 20 Points, 2 Dimensions ............................ 236
Figure 94: Maxmin Latin Hypercube Design, 80 Points, 3 Dimensions ............................ 237
Figure 95: Symmetric Latin Hypercube Design, 20 Points, 2 Dimensions ........................ 238
Figure 96: Symmetric Latin Hypercube Design, 21 Points, 2 Dimensions ........................ 239
Figure 97: Symmetric Latin Hypercube Design, 80 Points, 3 Dimensions ........................ 240
Figure 98: Visualizations of the Slope Problem ............................................................. 242
Figure 99: Visualizations of the Hyper-J Problem ............................................................ 244
Figure 100: Visualizations of the Newsstand Problem ....................................................... 247
Figure 101: Visualizations of the Queuing Network Problem ............................................. 249
LIST OF EQUATIONS

Equation 1 ........................................................................................................... 15
Equation 2 ........................................................................................................... 16
Equation 3 ........................................................................................................... 16
Equation 4 ........................................................................................................... 17
Equation 5 ........................................................................................................... 17
Equation 6 ........................................................................................................... 17
Equation 7 ........................................................................................................... 18
Equation 8 ........................................................................................................... 18
Equation 9 ........................................................................................................... 19
Equation 10 ........................................................................................................ 19
Equation 11 ........................................................................................................ 23
Equation 12 ........................................................................................................ 25
Equation 13 ........................................................................................................ 49
Equation 14 ........................................................................................................ 78
Equation 15 ........................................................................................................ 79
Equation 16 ...................................................................................................... 162
# Lists of Symbols and Abbreviations

A bold character such as \( x, \lambda, \) or \( A \) signifies a vector or matrix, whereas a non-bold character \((x_i \text{ or } y_j)\) is a scalar.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mu )</td>
<td>The real world response which we would like to know more about</td>
</tr>
<tr>
<td>( \phi(x) )</td>
<td>The function which relates real-world factors to the value of ( \mu )</td>
</tr>
<tr>
<td>( x_{rw}, [x^1, \ldots, x^q] )</td>
<td>The real world factors (controllable and not) that determine the value of ( \mu )</td>
</tr>
<tr>
<td>( q )</td>
<td>The number of factors (controllable and not) that determine the value of ( \mu )</td>
</tr>
<tr>
<td>( y )</td>
<td>The simulation model’s estimation of the response ( \mu )</td>
</tr>
<tr>
<td>( f(x, r) )</td>
<td>The function which relates simulation inputs to the value of ( y )</td>
</tr>
<tr>
<td>( x, [x_1, \ldots, x_d] )</td>
<td>The experimental simulation inputs</td>
</tr>
<tr>
<td>( y_i )</td>
<td>The simulation model’s estimation of the response ( \mu ) at point ( x_i )</td>
</tr>
<tr>
<td>( d )</td>
<td>The number of experimental simulation inputs</td>
</tr>
<tr>
<td>( r )</td>
<td>The random number stream which is utilized in stochastic simulation</td>
</tr>
<tr>
<td>( \hat{y} )</td>
<td>The simulation metamodel’s estimation of the response ( \mu )</td>
</tr>
<tr>
<td>( \hat{C}_{ij} )</td>
<td>The covariance between ( y_i ) and ( y_j ) also written as ( C(y_i, y_j) )</td>
</tr>
<tr>
<td>( \sigma^2 )</td>
<td>The variance of the data, for a generic point, ( y_i )</td>
</tr>
<tr>
<td>( x_0 )</td>
<td>The point which we would like a prediction at from our metamodel</td>
</tr>
<tr>
<td>( \hat{y}_0 )</td>
<td>The prediction at ( x_0 ) from the metamodel</td>
</tr>
<tr>
<td>( n )</td>
<td>Number design points for an experiment</td>
</tr>
<tr>
<td>( n_r )</td>
<td>Number of replications at a design point</td>
</tr>
<tr>
<td>( \mathcal{X} )</td>
<td>Design space for the simulation (set of all possible inputs)</td>
</tr>
<tr>
<td>( n_{cent} )</td>
<td>Number of center point is Central Composite Design</td>
</tr>
<tr>
<td>( N_{TP} )</td>
<td>Number of test points to be predicted at for testing metamodels</td>
</tr>
<tr>
<td>( C )</td>
<td>Covariance matrix (for kriging)</td>
</tr>
<tr>
<td>( C_{aug} )</td>
<td>Augmented Covariance matrix (for kriging)</td>
</tr>
<tr>
<td>Abrev.</td>
<td>Meaning</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>SM</td>
<td>Simulation Metamodeling</td>
</tr>
<tr>
<td>RSM</td>
<td>Response Surface Methods</td>
</tr>
<tr>
<td>MARS</td>
<td>Multivariate Adaptive Regression Splines</td>
</tr>
<tr>
<td>RBF</td>
<td>Radial Basis Functions</td>
</tr>
<tr>
<td>ANN</td>
<td>Artificial Neural Networks</td>
</tr>
<tr>
<td>SVR</td>
<td>Support Vector Regression</td>
</tr>
<tr>
<td>OR</td>
<td>Operations Research</td>
</tr>
<tr>
<td>OK</td>
<td>Ordinary Kriging</td>
</tr>
<tr>
<td>UK</td>
<td>Universal Kriging</td>
</tr>
<tr>
<td>SK</td>
<td>Stochastic Kriging</td>
</tr>
<tr>
<td>DOE</td>
<td>Design of Experiments</td>
</tr>
<tr>
<td>FFD</td>
<td>Fractional or Full Factorial Design</td>
</tr>
<tr>
<td>CCD</td>
<td>Central Composite Design</td>
</tr>
<tr>
<td>NTM</td>
<td>Number Theoretic Methods</td>
</tr>
<tr>
<td>CHD</td>
<td>Coffee House Design</td>
</tr>
<tr>
<td>RLH</td>
<td>Random Latin Hypercube Design</td>
</tr>
<tr>
<td>MMLH</td>
<td>Maxmin Latin Hypercube Design</td>
</tr>
<tr>
<td>SLH</td>
<td>Symmetric Latin Hypercube Design</td>
</tr>
<tr>
<td>SL</td>
<td>Slope Testbed Problem</td>
</tr>
<tr>
<td>HJ</td>
<td>Hyper-J Testbed Problem</td>
</tr>
<tr>
<td>NS</td>
<td>Newsstand Testbed Problem</td>
</tr>
<tr>
<td>QN</td>
<td>Queuing Network Testbed Problem</td>
</tr>
<tr>
<td>MAE</td>
<td>Mean Absolute Error</td>
</tr>
<tr>
<td>ME</td>
<td>Maximum Error</td>
</tr>
<tr>
<td>AC</td>
<td>Alpha Criteria</td>
</tr>
<tr>
<td>RD</td>
<td>Ranking Dominance</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>SD</td>
<td>Strict Dominance</td>
</tr>
<tr>
<td>WD</td>
<td>Weak Dominance</td>
</tr>
<tr>
<td>RRK</td>
<td>Reduced Rank Kriging</td>
</tr>
<tr>
<td>SVD</td>
<td>Singular Value Decomposition</td>
</tr>
<tr>
<td>RSVD</td>
<td>Reduced Singular Value Decomposition</td>
</tr>
<tr>
<td>ORK</td>
<td>Omni-Rank Kriging</td>
</tr>
<tr>
<td>CV</td>
<td>Cross Validation</td>
</tr>
<tr>
<td>OFR</td>
<td>Optimal Fixed Rank</td>
</tr>
<tr>
<td>OFRP</td>
<td>Optimal Fixed Rank Performance</td>
</tr>
<tr>
<td>IPORP</td>
<td>Individual Problem Optimal Rank Performance</td>
</tr>
<tr>
<td>CLT</td>
<td>Central Limit Theorem</td>
</tr>
<tr>
<td>BF</td>
<td>Brute Force</td>
</tr>
<tr>
<td>GA</td>
<td>Genetic Algorithm</td>
</tr>
<tr>
<td>o.d.e.</td>
<td>Optimal Design of Experiments</td>
</tr>
<tr>
<td>LHD</td>
<td>Latin Hypercube Design</td>
</tr>
<tr>
<td>OA</td>
<td>Orthogonal Array</td>
</tr>
</tbody>
</table>
CHAPTER 1: INTRODUCTION

1.1 Dissertation Statement

This dissertation is expected to improve an analyst’s use of simulations by offering improvements in the utilization of kriging metamodels for simulation analysis. There are three main contributions:

- An analysis of what comprises good experimental designs for practical (non-toy) problems when a kriging metamodel is used for analysis.
- An explanation and demonstration of how reduced rank decompositions can improve the performance of kriging as a tool for metamodeling.
- Development of a methodology, omni-rank kriging, which solves the problem of finding the optimal rank for use in the section before it.

1.2 Motivation

Because simulation allows greater fidelity in modeling than many other techniques, it is widely used. However, this greater fidelity comes at a computational price. Large, complex, industry-sized simulations require significant computation. Running a single replication may take hours, even days. In addition, simulation experiments frequently need multiple replications at each design point, and often, a number of distinct points need to be simulated to answer a question. These computational requirements cause these experiments to take unreasonably long times to run.
One of the preferred methods to circumvent these computational difficulties is simulation metamodeling (SM). SM is the process of taking data from the full simulation and then fitting a model to that data. In some sense, we can view a simulation model as simply a black box which gives us outputs for given inputs. The model we fit with SM is another black box, but it allows us to get output for given inputs computationally cheaper than from the full simulation. This faster model allows us to get information, particularly answers to optimization and sensitivity analysis questions, more quickly than if we used the full simulation. It is important to remember that after we are done using our simulation metamodel we should try to confirm our findings using the full simulation, if possible. Happily, this verification process should be eased by the knowledge gained from the metamodel’s behavior. The ability to analyze simulations which would otherwise be impossible to experiment on is the main motivation for SM. As a consequence, our motivation is improving the usage of SM.

1.3 Literature Review

This section will cover the introductory literature, and serve as a basis for all future work. The review has three parts. First, a brief description of simulation is given. This presentation will suggest that simulation is a key tool for analyzing real problems with depth. The second part of the review will concentrate on SM. This presentation will cover the existing research in SM methods. After describing various methods, we will then present some of the successful examples of using SM. The third part of the review will present some basic background on spatial statistics and, in particular, the types of kriging. This
presentation of kriging will provide the technical details of the method which is the main subject of the dissertation.

1.3.1 Literature Review on Simulations

Simulations are generally created so that we can learn more about a real-world system. While there are many valid (and approximately interchangeable) ways to view simulations, I subscribe to the views given by Law and Friedman (Friedman, 1996; Law, 2006). In their view, simulations are models of real world systems [Note: the following description of how simulations fit into science closely follows Section 1.2 of Law (2006)]. Specifically this means that simulations have a few characteristics which distinguish them from other models of those real-world systems. Figure 1 shows the hierarchy in deriving the simulation model from a real-world system.
When we wish to learn about the behavior of a real-world system, there are a variety of methods that can be used. First, we must decide if we want learn about the behavior (response of interest) of the system directly from the actual system, or a model of that system. Working with a model of the simulation has the benefit of finding out how the real-world system is likely to behave without actually running these experiments in the real world (which may be costly, unsafe, unethical, and/or time consuming). However, choosing to work with a model has the risk that our model may not adequately reproduce the behavior of the true system. This danger is real, but over time, analysts have become adept at ensuring that their models do produce highly accurate results. Further, there have been decades (in fact

Figure 1: Modeling Hierarchy
centuries) of mounting evidence that making models of complex phenomena is an effective way of analyzing the real-world systems.

Second, given we decide to build a model of the real-world system, will we build a physical model or a mathematical one? Physical models can be useful in a variety of situations. They are used in design of vehicles, physics, chemistry and a plethora of material sciences. However, there are a variety of situations where a physical model would be impossible or impractical to build. These situations can arise for a variety of reasons such as:

- The formation of solar system, which it is impossible to simulate physically;
- A human healthcare system, which may be unethical to experiment on; and
- A plane crash, which would be too costly to try repeatedly (or probably even once).

This dissertation will broadly consider mathematical models.

Third, given we have decided to build a mathematical model, we must decide if it will be an analytical or simulation model. We normally prefer to create an analytical model when possible. Analytical models frequently can be “solved” such that the behavior of the system as a function of its parameters can be understood mathematically. Having an analytic model allows us to characterize important model results such as inverses, optimization, and sensitivity analysis. Further, a simulation model tends to include a number of computational difficulties not needed for analytical ones (for example, warm up analysis and output analysis). However, we frequently find that creating analytic models for complex problems leads to difficulties with computing solutions. Often solving these models can require
convolutions and/or inverses of matrices which employ numerical calculations, which diminishes the advantages of creating analytical models in the first place. Further, simulations allow us to incorporate a plethora of system behavior which would cause our analytic models to be intractable.

So, the use of simulation models is very popular. There are still more factors which distinguish some simulations from each other. The research in this dissertation is generally applicable to most simulation types: dynamic and static, continuous and discrete, and stochastic and deterministic.

Another factor distinguishing different simulations is the number of responses of interest. We will generally proceed under the assumption that there is only one response of interest. While this is likely not true in practice, the concepts presented in this dissertation may be generalizable to multiple responses. Kleijnen and Mehdad (2012) have applied cokriging (a modification of Kriging for multiple responses) to the problem of multiple responses. In Table 1 we introduce the notation for simulation that will be used for the rest of this dissertation [Note: this is mainly the notation used by Friedman (Friedman, 1996)].
Table 1: Model Notation

<table>
<thead>
<tr>
<th>Real-world System</th>
<th>Simulation Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model $\mu = \phi(x^1, x^2, \ldots, x^d)$</td>
<td>$y_i = f(x^1, x^2, \ldots, x^d, r_i)$, $i = 1, \ldots, n_r$</td>
</tr>
</tbody>
</table>

Where:
- $\mu$ is the response of interest
- $q$ is the number of factors (controllable and not) that determine the value of the response
- $y_i$ is the simulation response in the $i$th replication
- $n_r$ is the number of replications at a design point
- $d$ is the number ($d < q$) of controllable inputs
- $x_j$ is the value of the $j$th experimental input factor
- $r_i$ is the vector of random numbers/variates used in the $i$th replication

$r_i$ represents all the $q - d$ uncontrollable factors (normally represented as random variables) from the real system in the simulation model.

The writing on the practice and use of simulation is extensive. The text by Law (2006) is a good general reference covering a wide range of simulation-related topics, not specifically relevant to this dissertation.

1.3.2 Literature Review on Metamodelling

As computing power has grown, systems of greater complexity may be simulated. However, the size of simulations has generally kept pace with the increase in computation power (Koch, 1999). This growing size of simulations has meant that computational constraints have remained vexing, and so researchers have sought ways to address these difficulties. One method which has been popular and fruitful is metamodelling.

In metamodelling we seek to “model our model.” Just like our simulation model is an abstraction of reality, we attempt to build a model which is an abstraction of our simulation (J. P. C. Kleijnen, 1975). Sometimes we use metamodelling to achieve greater understanding of the simulation model. But for the purposes of this dissertation, we plan on using our
metamodel for analysis; so the metamodel should have two key properties to make this effort worthwhile. First, it should be a highly accurate representation of the simulation model. That is, we want our metamodel to produce outputs similar to the outputs that the original simulation would produce given the same inputs. Second, we want the metamodel to be computationally faster. Specifically, we want to reduce the computer time between input and output. Wang and Shan (2007) give a good survey article on how metamodeling assists in engineering design. They give references which cover a multitude of aspects including: design of experiments, metamodeling techniques, metamodel evaluation, and a variety of applications.
A relationship among the real-world system, the simulation model, and the metamodel as black box functions is shown in Figure 2.

![Diagram of relationship between models]

Each level to the right in Figure 2 represents greater abstraction and less fidelity. It should mean greater ease of experimentation and speed of the turnaround time from input to output. On the other hand, each level to the right also represents greater risk that we will lose the validity of our input/output relationship.
There are many potential methods for constructing a metamodel. Barton (2009) gives seven different methodologies in his 2009 paper on SM for optimization. He lists: linear/quadratic regression, higher order regression, nonlinear regression, radial basis function, neural network, spatial correlation (kriging), and spline metamodels. There are other possible metamodels (ex. support vector regression), but the previous cover the vast majority of choices offered.

Response surface methods (RSM or ordinary regression) are not the most active metamodeling research area, but it is still a reasonable preliminary choice due to its simplicity and the fact it is based on generally understood statistical methodology. Tunali and Batmaz (2000) give an example of metamodeling for a simulation of a time shared computer system using regression and then confirm the appropriateness of their model. Aytug et al. (1996) also use regression; their application is to answer questions about a Just-In-Time manufacturing system, specifically how many kanbans should be employed. Yang et al. (2010) use nonlinear regression and an interesting progressive fitting technique to model the tradeoffs in production of semiconductor manufacturing.

Spline interpolation (often called multivariate adaptive regression splines and referred to as MARS) is also well known, though it is not as widely used as regression. Splines use localized functions to reduce the risk of bad global fitting which frequently occurs with standard regression (Runge’s phenomena). Keyes and Rees (2004) do a thorough job investigating the appropriateness of splines for metamodeling for optimization.
Radial basis functions (RBF) are a standardized way of creating weights to place on the data to interpolate at points where there is no information. They are quite popular due to their simplicity and robustness. Hussain et al. (2002) compare the performance of RBF to standard regression and find it to be superior. Shin et al. (2002) give a systematic way to balance the underfitting and overfitting using RBF. Mullur and Messac (2006) present what they call “extended RBF” and show it to have great potential.

The results on using artificial neural networks (ANN) for SM are not as encouraging. Generally, the findings seem to show that the method’s results are passable and using ANN properly requires significant analyst skill. Sabuncuoglu and Touhami (2002) describe many of the issues in using ANN in their paper which focuses on a simulation of a job shop. They find it performs admirably for some tasks, but not all, and there are many factors which influence those behaviors. Fonseca and Navaresse (2002) and Fornseca et al. (2003) give an example of using ANN to model another job shop problem. They show it to be an acceptable alternative to using conventional software, probably trading accuracy for speed. Talbach et al. (2007) use ANN for metamodeling in a simulation of groundwater contamination in a road project to much success. Kuo et al. (2007) examined one method for how limited data issues can be approached while using an ANN, a very important problem for SM.

Kriging will receive more attention in this dissertation; here we focus on giving a few examples of its application to simulation metamodeling. However, we first must mention Kleijnen’s (2009) excellent review article on kriging for SM. It also mentions (directly or indirectly) as future work the topics addressed in this dissertation. Biles et al. (2007) use
kriging to assist in the optimization of an (s, S) inventory model. In a more complex application, Booker et al. (1999) use kriging for the optimization of a helicopter’s rotor design. They primarily work on what optimization method works best, using a kriging surrogate model. Huang et al. (2006) incorporate kriging metamodels into their optimization method and show that it is quite effective.

Support vector regression (SVR) is new technique for SM. Wang et al. (2010) use SVR and metamodeling to optimize vehicle crash worthiness to nice effect. Clark et al. (2005) give a useful comparison of SVR to a variety of methods (kriging, MARS, RBF, and RSM). They show it seems to be the best option for their standard engineering test-bed problems (though kriging is very close and no confidence intervals are given). Xiang and Huang (2010) employ SVR as their surrogate method in an optimization scheme and compare it to other metamodeling methods (ANN, kriging, and RSM). They show it to be an effective method and they further conclude that it is superior to all three other methods. However, this conclusion is undermined by Table 1 in their paper (kriging seems to be better, but there are no confidence limits so the difference may be insignificant).
There are a large number of papers which attempt to compare various metamodeling techniques (H. Fang, Rais-Rohani, Liu, & Horstemeyer, 2005; Ruichen Jin, Chen, & Simpson, 2001; Y. F. Li, S. H. Ng, M. Xie, & T. N. Goh, 2010; Sakata, Ashida, & Zako, 2003; Villa-Vialaneix, Follador, Ratto, & Leip, 2012). The comparisons usually include the following elements:

- Introduce a set of metamodeling methods to be tested;
- Optional: give more information about a (some) method(s) which is less frequently used or new (many of the papers listed previously have this property);
- State what experimental design will be used, (often very slight coverage);
- Explain the test-bed set of problems that each method will attempt to metamodel, and the parameters of each problem (signal/noise ratio, number of design points, etc.);
- Explain the metrics for the comparison, most commonly some subset of the following:
  - A measure of average error, mean absolute error or root mean square error;
  - A measure of maximum error, max absolute error;
  - $R^2$, a measure of the fit quality between the metamodel and test data;
- Employ cross-validation procedures to validate the metamodels; and
- Draw conclusions about what methods seem to be best when, and possibly, why.

Simpson et al. (2001) give a review of deterministic applications and Jin et al. (2001) consider the effects of stochastic noise in addition to deterministic applications (albeit only slightly).
1.3.3 Literature Review on Spatial Statistics and Kriging

Kriging was developed for use in mining and geostatistics by Krige (1951) and Matheron (1963). The objective of kriging is to estimate the data response value, \( y_0 \), at a new point which does not already have data collected, \( x_0 \), using the data \( x \) and \( y \). The point \( x_0 \) is a combination of input values for which we are interested in estimating the simulation the response without needing to run the simulation. In the following section we give a short derivation of kriging. This derivation of kriging is presented because there are aspects of it which will be referred to later in this dissertation.

There are a set of assumptions which must be introduced and then given some support. First, we must specify a covariance function, \( C(y_i, y_j) \) (also written as \( \hat{C}_{ij} \)), which describes the relationship between points in the data set. This relationship is often very difficult to ascertain with great certainty from the sample data (N. A. C. Cressie, 1993). Fortunately, this problem does not seem to undermine the methodology (N. A. C. Cressie, 1993). For more information on covariance functions and their relationships to variograms and kriging we refer to Cressie’s (1993) oft cited text and Issak’s (1989a) more intuitive one.

We must make a further assumption about the data, namely it is stationary. This means that the expected value for data in a generic area of the data set is the same in any other generic area, subject to the caveat that the two areas are far enough away that the covariance between the two is negligible. In other words, there is no discernible large scale trend. However, we know this not to be true in most of our situations. There are two justifications for why this is not a reason we cannot continue. First, de-trending or using
universal kriging (more on the different types of kriging later) can manipulate the data so this assumption is met (J. P. Kleijnen & W. van Beers, 2005; Jay D Martin & Simpson). Second, there has been evidence that even though large scale trend is present, using this stationarity assumption can still be successful in modeling (Ankenman, Nelson, & Staum, 2010; Beers & Kleijnen, 2003).

Besides the assumptions about the data (that a covariance function can be found which describes it well and stationarity), there are three additional assumptions which make the kriging model tractable. The first of those three assumptions is linearity. As is common notation, \( \hat{y}_0 \) is the name for our model’s prediction of \( y_0 \). Linearity means that the model we will use is a weighted combination of the data points. This form is shown in the Equation 1 below.

\[
\hat{y}_0 = \sum_{i=1}^{n} \lambda_i y_i. \tag{Equation 1}
\]

The key question based on choosing this linear restriction is how to pick those values of \( \lambda_i \) given their associated locations \( x_i \) for \( i = 1, ..., n \). We have two other assumptions which will provide further assistance.

The next assumption (our fourth overall so far) is unbiasedness. Unbiasedness means that we want to be able to expect our predictions to be too high and low equally. In particular, this means that the expectation of the residuals of the predictions should be zero, which is shown in the equation below,

\[
E[\hat{y}_0 - y_0] = 0.
\]
Since we have already assumed linearity in the predictor model, we get the following:

\[
0 = E[\hat{y}_0 - y_0] = E \left[ \sum_{i=1}^{n} \lambda_i y_i - y_0 \right] \rightarrow E \left[ \sum_{i=1}^{n} \lambda_i y_i \right] = E[y_0] \rightarrow \sum_{i=1}^{n} \lambda_i E[y_i] = E[y_0]
\]

Since we also have made the stationarity assumption, we then get:

\[
\sum_{i=1}^{n} \lambda_i E[y_i] = E[y_0] \rightarrow \sum_{i=1}^{n} \lambda_i = E[y] \rightarrow \sum_{i=1}^{n} \lambda_i = 1
\]

\[
\sum_{i=1}^{n} \lambda_i = 1. \quad \text{Equation 2}
\]

Our final assumption is that we are looking for the best possible (optimal) predictor. In a mathematical sense this means we try to minimize the size of the residuals. This objective is best accomplished by minimizing the variance of the errors. In total, we are seeking the weights, \( \lambda_i \), which will minimize \( Var(\hat{y}_0 - y_0) \), subject to the assumptions already stated (known covariance function, stationarity, model form, and unbiasedness).

Now we set formulate the minimization problem. This problem is cast as a constrained minimization, a common problem in operations research (OR). We choose to solve it via Lagrange multipliers, and so take a constrained optimization and recast it as an unconstrained one. After doing so, solving the resultant problem is straightforward, and gives us a solution which works for the original one,

\[
Var(\hat{y}_0 - y_0) = Cov(\hat{y}_0, \hat{y}_0) - 2Cov(\hat{y}_0, y_0) + Cov(y_0, y_0). \quad \text{Equation 3}
\]
We will study the three parts of the right hand side of Equation 3 separately. First the $\text{Cov}(\hat{y}_0, \hat{y}_0)$ part:

$$\text{Cov}(\hat{y}_0, \hat{y}_0) = \text{Var}(\hat{y}_0, \hat{y}_0) = \text{Var}\left(\sum_{i=1}^{n} \lambda_i y_i \right) = \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j \hat{C}_{ij}. \quad \text{Equation 4}$$

Second the $\text{Cov}(y_0, y_0)$ part:

$$\text{Cov}(y_0, y_0) = \sigma^2. \quad \text{Equation 5}$$

Lastly, the $2 \text{Cov}(\hat{y}_0, y_0)$ part:

$$2 \text{Cov}(\hat{y}_0, y_0) = 2 \text{Cov}\left(\sum_{i=1}^{n} \lambda_i y_i , y_0 \right) = 2 \left( E\left[\left(\sum_{i=1}^{n} \lambda_i y_i \right) \cdot y_0 \right] - E\left[\sum_{i=1}^{n} \lambda_i y_i \right] \cdot E\left[y_0 \right]\right) =$$

$$2 \left( \sum_{i=1}^{n} \lambda_i \cdot E[y_i \cdot y_0] - \sum_{i=1}^{n} \lambda_i \cdot E[y_i] \cdot E[y_0] \right) =$$

$$2 \sum_{i=1}^{n} \lambda_i \cdot (E[y_i \cdot y_0] - E[y_i] \cdot E[y_0]) = 2 \sum_{i=1}^{n} \lambda_i \cdot \text{Cov}(y_i, y_0) =$$

$$2 \sum_{i=1}^{n} \lambda_i \hat{C}_{i0}. \quad \text{Equation 6}$$
Substituting the results of Equations 4, 5, and 6 back into the expression for the variance (Equation 3):

\[ \text{Var}(\hat{y}_0 - y_0) = \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j \hat{C}_{ij} - 2 \sum_{i=1}^{n} \lambda_i \hat{C}_{i0} + \sigma^2. \]  
Equation 7

Equation 7 is what we want to minimize with respect to the weights \( \lambda \), subject to the unbiasedness condition \( \sum_{i=1}^{n} \lambda_i = 1 \) (Equation 2). Since the \( \sigma^2 \) is fixed, we can drop it from the function we are trying to minimize. So our augmented objective function (Lagrangian) is

\[ L(\lambda, \mu) = \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j \hat{C}_{ij} - 2 \sum_{i=1}^{n} \lambda_i \hat{C}_{i0} + 2\mu \left( \sum_{i=1}^{n} \lambda_i - 1 \right). \]  
Equation 8

Now we take partial derivatives of Equation 8 with respect to \( \lambda_i \)’s and \( \mu \).

\[ \frac{\partial L(\lambda, \mu)}{\partial \lambda_i} = 2 \sum_{j \neq i}^{n} \lambda_j \hat{C}_{ij} + 2\lambda_i \hat{C}_{ii} - 2\hat{C}_{i0} - 2\mu = 2 \sum_{j=1}^{n} \lambda_j \hat{C}_{ij} - 2\hat{C}_{i0} + 2\mu, \]

\[ \frac{\partial L(\lambda, \mu)}{\partial \mu} = 2 \left( \sum_{i=1}^{n} \lambda_i - 1 \right). \]
Setting all the partial derivatives to equal 0 yields the following.

\[ 0 = 2 \sum_{j=1}^{n} \lambda_j \hat{c}_{ij} - 2 \hat{c}_{i0} + 2 \mu \rightarrow \hat{c}_{i0} = \sum_{j=1}^{n} \lambda_j \hat{c}_{ij} + \mu, \]

\[ 0 = 1 - \sum_{i=1}^{n} \lambda_i \rightarrow 1 = \sum_{i=1}^{n} \lambda_i. \]

These \( n + 1 \) equations are all linear in \( \lambda_i \) and \( \mu \), and so can be rewritten in matrix form, which is shown in Equation 9,

\[
\begin{bmatrix}
\hat{c}_{11} & \hat{c}_{12} & \cdots & \hat{c}_{1n} & 1 \\
\hat{c}_{21} & \hat{c}_{22} & \cdots & \hat{c}_{2n} & 1 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
\hat{c}_{n1} & \hat{c}_{n2} & \cdots & \hat{c}_{nn} & 1 \\
1 & 1 & \cdots & 1 & 0
\end{bmatrix} \begin{bmatrix}
\lambda_1 \\
\lambda_2 \\
\vdots \\
\lambda_n \\
\mu
\end{bmatrix} = \begin{bmatrix}
\hat{c}_{10} \\
\hat{c}_{20} \\
\vdots \\
\hat{c}_{n0} \\
1
\end{bmatrix}. \quad \text{Equation 9}
\]

Equation 9 is a linear system of equations and commonly referred to as the ordinary kriging equations. This simple linear system can be solved in a variety of ways, direct inversion, Gaussian elimination, or decomposition methods to name a few (the form of this solution is the main subject for the second part of the dissertation). With the \( \lambda_i \) values solved for from Equation 9, we apply them to the data values we have and obtain our estimate, \( \hat{y}_0 \).

In Equation 10 we show how the solving (via obtaining the inverse) of Equation 9 can be expressed in a closed form.

\[
\begin{bmatrix}
\lambda \\
\mu
\end{bmatrix} = C^{-1}_{aug} c_0 \rightarrow \hat{y}_0 = \lambda^T y. \quad \text{Equation 10}
\]
Where $\mathbf{c}_{aug}$ is the augmented (by ones and a zero) matrix of covariances from Equation 9, $\mathbf{c}_0$ is the augmented (by a one) vector of covariances between $x_0$ to the data, and $\mathbf{y}$ is the data, $\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$.

There are two common types of kriging with respect to how they model trend in the data, ordinary and universal. The difference between the two is best shown by looking at how each utilizes the same decomposition for the modeling the data. Kriging views the data as being comprised of three parts as shown in the equation below.

$$Z(\mathbf{x}) = T(\mathbf{x}) + W(\mathbf{x}) + \epsilon(\mathbf{x})$$

Where:

- $Z(\mathbf{x})$ is the data value at the point $\mathbf{x}$, $y_\mathbf{x}$ in our earlier notation.
- $T(\cdot)$ is equal to $E(Z(\cdot))$ and is the overall deterministic trend.
- $W(\cdot)$ is a zero-mean, $L_2$-continuous (i.e., $E(W(s + h) - W(s))^2 \to 0$ as $\|h\| \to 0$), intrinsically stationary process.
- $\epsilon(\cdot)$ is a zero-mean white-noise process, independent of $T$ and $W$.

In ordinary kriging (OK), we assume $\mu(\mathbf{x}) = \mu$. This means there is some unknown mean which we will estimate from the data. This assumption is sometimes sufficient for good modeling and is frequently used in literature. In universal kriging (UK), we assume $\mu(\mathbf{x}) = \Sigma_{i=1}^b \beta_i q_i(\mathbf{x})$. Thus, UK allows us to add regression functions. While it does add some
computation, it allows for substantially improved modeling. UK sometimes goes by other names in the OR literature (detrended kriging is nearly the same thing (Beers & Kleijnen, 2003; J.D. Martin & Simpson, 2003)) and serves as the basis for some more advanced methods (Ankenman et al., 2010; Staum, 2009).

For another excellent and intuitive derivation of OK we refer to Isaaks' text’s Chapter 12 (Isaaks, 1989b). For a more compact and technical derivation we refer to Cressie text’s Chapter 3, which gives derivations for both OK and UK (N. A. C. Cressie, 1993).

It is important to distinguish here that what we just derived is for OK and not UK. We chose to show how OK is derived for a couple of reasons. First, OK’s derivation is simpler than UK’s and so can be presented without as much length and is easier to understand. Second, the computation aspects we will utilize will only need the parts shown (which are shared by both OK and UK).

The recent increase in the OR literature using kriging can be traced to three main reasons. First, the average analyst has increased computing power. Kriging is a computationally intensive process (relative to most other metamodeling techniques) since it requires inversion of a covariance matrix. While this is not a huge cost, to find the appropriate covariance model, most references recommend using maximum likelihood estimation of the covariance parameters (Booker et al., 1999; N. A. C. Cressie, 1993; Isaaks, 1989a; Martin & Simpson, 2005). This computation then requires optimization of a function which needs (approximately) inversion of an \( n \times n \) matrix (where \( n \) is the number of data points) for each function evaluation. This computation becomes a significant computational
burden for large problems. However in situations with limited data, like those requiring metamodeling, this cost is not significant within the environment of today’s powerful desktop computers. A decade or two ago, this cost could have been prohibitive, and so Kriging has become more popular recently.

A second reason for the increased interest in kriging as a metamodeling technique is that it is rather robust to misspecification problems. There have been a variety of papers to demonstrate the hazards of metamodel misspecification (Ankenman et al., 2010; J. P. C. Kleijnen & W. C. M. van Beers, 2005; Reis dos Santos & Porta Nova, 2006; Villa-Vialaneix et al., 2012). Misspecification occurs when the regression form picked does not adequately reflect the true underlying process and so the model fitted will not be effective for prediction. This is further worrying because it is possible to achieve good performance on measures of fit with a misspecified model which will then be poor for predicting new points. Kriging does not suffer from the same hazards since it is an interpolation method which uses the average of the data to supply missing information. This approach means it has been found to be robust in most circumstances (J. P. C. Kleijnen & W. C. M. van Beers, 2005).

A third reason for greater interest in kriging is that it is a rather flexible technique. As simulation applications have grown, there has been a need for techniques to model a wider variety of data sets. Since there are a variety of choices for the spatial correlation model, there are a great number of data sets which can be fit very well by a kriging metamodel. Beyond the functional form picked, kriging does not have to treat all directions equally. By making use of “anisotropic” correlation functions, an analyst can model behavior in each
direction. This observation means if the response varies differently (specifically differing levels of sensitivity) with respect to different inputs we can properly account for that difference.

1.3.4 Stochastic Kriging

Stochastic kriging (SK) is a recent modification of Kriging created by operations researchers for use in OR (Ankenman et al., 2010; Staum, 2009). SK utilizes information which is often available in OR simulations and is not present in most geostatistical applications. In particular they break the uncertainty of predictions down into two sources: intrinsic and extrinsic, which will be explained below. By doing their analysis in this way they are able to achieve significant improvements over standard Kriging (Staum, 2009).

To understand how the division of uncertainty arises we must first review their (SK developers) approach to modeling stochastic data. First they consider metamodeling data *without* stochastic noise. To model such data the response model (universal kriging) has been shown to be effective. In Equation 11 we show the form of universal kriging,

\[
\hat{Y}(x) = f(x)^T \beta + M(x).
\]  

Equation 11

In this model we see the overall trend \((f(x)^T \beta)\) and random field \((M(x))\) are present. The random field portion is a single realization of a mean 0 random field. Specifically, \(M\) is a single realization, randomly sampled from a space of functions mapping \(\Re^d \rightarrow \Re\). Further, it is assumed that the functions in this space exhibit spatial correlation, implying that values
$M(x)$ and $M(x')$ will tend to be similar if $x$ and $x'$ are close to each other. The uncertainty due to the stochastic nature of $M(x)$ is termed extrinsic “because it is imposed on the problem (not intrinsic to it) to aid in developing a metamodel” (Ankenman et al., 2010).

Next, they then take this base model and augment it for use with data with noise as shown below:

$$\hat{Y}_j(x) = f(x)^T \beta + M(x) + e_j(x),$$

where $e_j(x)$ is the intrinsic noise for each replication at a design point, $x$. In OR this intrinsic uncertainty normally comes from the differences in the random variates due to independent random number streams being used for independent replications. This is noise which is inherent to the problem, and no metamodeling method would be immune to it, hence the name intrinsic uncertainty is used.

The breakdown of uncertainty into intrinsic and extrinsic uncertainty is not only insightful, but it can be leveraged to create better predictions. In the previous section we mentioned that kriging had already approached the problem of measurement error (the geostatistical equivalent for intrinsic uncertainty). The standard approach in kriging is to then estimate each parameter via MLE without further deliberation. This is where SK truly diverges from the previous kriging methods.

In the simulation experiments we encounter in OR, we often have good estimates of precisely the magnitude of the stochastic noise that is present at each individual point in our data. This means that if we choose to decompose the covariance matrix into the extrinsic and
intrinsic parts we may be able to get better predictions. Specifically, where OK uses Equation 10, SK uses

$$[\lambda] = [C_M + C_e]^{-1}c_0 \rightarrow \hat{y}_0 = \lambda^T y,$$

Equation 12

where, as outlined in Ankenman et al. (2010), $C_M$ is the covariance matrix found via MLE on the data (extrinsic) and $C_e$ is a covariance matrix found from analysis of the simulation replication output (intrinsic).

This improvement is possibly due to making the MLE estimation simpler (fewer degrees of freedom and hence less opportunity for multiple wrong parameter values to conceal each other). Phenomena such as this is have also been found by Kleijnen (2008; J. P. C. Kleijnen, 2009).

Beyond the preliminary proposal papers of SK there have been other recent uses. First, there have been more theoretical papers which seek to improve the use of SK. Topics covered in those papers are utilizing gradient information (Chen, Ankenman, & Nelson, 2013) and the effects of common random numbers in simulation experiments (Chen, Ankenman, & Nelson, 2012). There are also a set of papers showing application of the SK to financial problems (Baysal, Nelson, & Staum, 2008; Liu & Staum, 2010) and another working paper at http://stochastickriging.net/.
1.4 Planned Contributions to Literature

As stated earlier there are three significant contributions to the current operations research (OR) literature from this dissertation. Each of the contributions will be focused on improving the analyst’s use of simulations via improving the utilization of Kriging metamodels for simulation analysis. In Figure 3 (shown on the next page) we show a visual representation of how decisions in data collection and data modeling (metamodeling) propagate into the final quality of predictions. In essence this dissertation will look at some of the issues in how best to make decisions in the flow.
Figure 3: Metamodeling Decision Visualization
The first contribution will be finding good experimental designs for practical problems when a kriging metamodel will be used for analysis. This analysis will advance the use of kriging in three main ways. First, we will test a wider variety of designs than have been tested together, providing for more comprehensive results. Second, we will test these designs on non-toy size problems. In particular we will test in situations with high dimension. Third, we will look at how various problem characteristics and design characteristics interact to affect the quality of our metamodels. The sum result of these findings will be information analysts can use to pick the best design ahead of time, and get the most out of their computation budgets. Referring to Figure 3, this explores the decision in the “data point placement box.”

The second contribution will show how effective use of a reduced rank decomposition can improve the performance of kriging as a tool for metamodeling. While here are some small computational gains to be had from this process in theory, the real gains come from getting better predictions from our metamodels. These better predictions come about because reduced rank kriging creates a tuning parameter (the rank used) which mimics the nugget effect which is commonly used to model data with stochastic noise. In some circumstances this turning allows us to better account for the noise present in the data. In Figure 3, this is studying the arrow from “\( C_{\text{aug}} \) Matrix” to “Inverse \( C_{\text{aug}} \) Matrix”.

The third contribution will introduce omni-rank kriging. This is an extension of reduced rank kriging intended to answer the question of how to find optimal or near optimal for use in reduced rank kriging. Omni-rank kriging uses all potential ranks, and so it does not
need to search for any particular rank. This modification realizes the potential gains from reduced rank kriging and provides a workable methodology for simulation metamodeling.
CHAPTER 2: EXPERIMENTAL DESIGN FOR SIMULATION
METAMODELING USING KRIGING

2.1 Abstract

Simulation metamodeling always attempts to extract maximum information from limited data. Sequential designs are generally known to be more efficient than “one shot” designs. However, sequential designs require some sort of pilot design from which the sequential stage(s) can then take over. We seek to find good designs for these pilot studies as well as designs which will be effective if there is no sequential stage to come. We test a wide variety of designs over a small set of test-bed problems. Our findings indicate that analysts should try to take advantage of any prior information they may have about their problem and their goals in metamodeling. In the event of a total lack of information we find that Latin hypercube designs are robust default choices. Our work is most distinguished by the higher levels of dimensionality we study.

2.2 Motivation

SM is a valuable technique for analyzing simulations. It allows analysts to save significant computational effort. These savings can allow for analysis of simulations which would be otherwise intractable. In situations where SM is valuable, good experimental design can be exceedingly valuable (Sanchez & Wan, 2009). This section will contribute to experimental design for SM in three main ways.
First, we will test a wider variety of designs than have been tested together, providing for more comprehensive results. Further, we will include more advanced designs than those that currently appear in literature (J. P. C. Kleijnen, 2009). Using this variety of designs will increase the knowledge about how design characteristics affect the ability to execute simulation experiments effectively.

The second improvement, is one which the current literature suggests for future work: extending beyond experience with one or two dimensions (J. P. C. Kleijnen, 2009). While small dimensionality examples are instructive, they do not recognize one of the major problems facing a simulation practitioner, the curse of dimensionality. One feature which makes simulation analyses particularly challenging is that a large number of input variables are often included in real-world simulations. Thus test problems with many, up to 30, dimensions gives our investigation greater robustness than is found in the current literature.

Third, this work will augment the current research by looking at how problem and design characteristics interact to affect the quality of our metamodels. Simply picking a test-bed of problems and giving a “best” design (as is common in the literature) gives analysts only limited guidance. However, giving information on what problem characteristics affect which designs perform best give greater guidance and insight into effective SM practice.
2.3 Literature Review on Experimental Design for Kriging

Design of Experiments (DOE) is the study of how best to select the inputs to be tested. As is common in DOE, we assume that there is some given design space (the area which is to be studied) named $\mathcal{X}$. The space $\mathcal{X} \subset \mathbb{R}^d$ is the full set of possible inputs into the simulation. A specific point in $\mathcal{X}$ corresponds to a combination of inputs. We then seek to place our $n$ points in $\mathcal{X}$ where they will do the most good for our simulation metamodel. We will define $\mathcal{X}$ more specifically (in the most common way) in a later section.

The previous work in experimental design for use in kriging draws from two major research streams. The first is the traditional experimental design literature on kriging in spatial statistics. The second is the OR literature on experimental design for SM generally, and kriging metamodels particularly.

2.3.1 Experimental Design in Statistics for Kriging

The quantity of literature on experiment design in statistics would be hard to overestimate (simply searching experimental design in Google Scholar returns over 4 million results). Good reference texts for anyone interested in general experimental design are the texts *Planning, Construction, and Statistical Analysis of Comparative Experiments* (Giesbrecht & Gumpertz, 2011) and *The Handbook of Statistics, volume 13* (S. Ghosh, 1996). However, we are only interested in the literature about experimental design for
kriging (a result which returns a “mere” 15 thousand results). We recommend the general text *Collecting Spatial Data* (Müller, 2007) for an introduction.

The main thrust of DOE in statistics with respect to kriging has focused on studying the effects of lattice designs on kriging’s predictions. These designs are quite simple, a repeating pattern in the experimental space. The most common lattice designs are grids. The three most common grids in literature are square, triangular, and hexagonal, with square being the most common among those. There have been a few papers which tried to distinguish which grid shape was best, but they generally show that the differences are slight, with different papers finding competing results (Olea, 1984; Yfantis, Flatman, & Behar, 1987). Also, it is important to note that these papers only study a two dimensional $\mathcal{X}$, so their applicability to our situations is likely limited.

The strength of these grid designs is that they cover the space of $\mathcal{X}$ intuitively (they keep points well away from each other). However, these designs are vulnerable to the curse of dimensionality and are not receptive to augmentation, both of which are problems for analysts (K. Fang, 1994). Finally, there are no good high dimensional representations of non-rectangular grids. It is for this reason we will proceed with only rectangular grids in our study.

There are a wide variety of other possible designs from the statistics community. In Appendix A we go into detail discussing different design theories. We detail how their ideas may or may not apply to our problem. In the section 2.4.1 we detail the particular designs selected for this dissertation study, why they were selected, and how we are implementing
them. The two introductory texts listed at the beginning of this section are good references to lead interested readers into wide array of other design options.

2.3.2 Experimental Design for SM in OR

The second major literature stream of interest to us is from OR and engineering more generally. This work is less formal, but more likely to be fruitful. We seek to improve on this literature generally by using both more advanced designs (from the statistics literature) and larger OR problems for testing.

There is a significant division in the kinds of designs that are used in OR simulations: “one-shot” designs and sequential designs. “One-shot” designs have the number simulation runs fixed in advance and then analysis proceeds. This contrasts with sequential design, where successive additional runs may be done with different input points which depend on the analysis from previous runs. All designs covered in Appendix A are fixed run size designs. While some statistics references consider sequential design, many applications are not good candidates for sequential design (agriculture, clinical trials, etc.). OR and engineering have more applications which are amenable to sequential design (production systems, serial computer experiments, and product design), hence sequential design receives more attention there.

In OR studies, sequential designs have been shown to require more effort in implementation, but also to have significant rewards. Sequential designs are shown to be
more effective on a point by point basis in several references (R. Jin, Chen, & Sudjianto, 2002; Kleijnen & Van Beers, 2004; W. C. M. van Beers & Kleijnen, 2004; Wim C. M. van Beers & Kleijnen, 2008). However, most sequential designs require some “exploratory” phase which is conducted as a fixed sample size design. Further, the level of skill in creating a sequential design is significant enough to be a barrier to entry for the average analyst. Hence we will proceed by working on the problem of picking a best fixed sample size design, either to complete a study or seed a sequential design.

Simpson et al. (2001) give a nice survey article covering many aspects of SM. They cover metamodeling techniques, experimental design, and how metamodeling fits into engineering design. They pay particular attention to how the established statistical techniques port over to deterministic models, for which they may not have been designed. Some of our work is based on their directions for future work, particularly our work with larger problems.

Kleijnen (1998) also gives a good overview of the issues facing an analyst. His book chapter covers both strategic and tactical issues intended to improve the usage of SM. He asserts that screening designs are an overlooked issue in DOE for SM. Screening is used to separate out the main factors driving the variation in simulation responses. He observes that once screening has reduced the number of experimental variables to 15 or less, standard design work can begin. In his coverage of standard design he focuses on fractional factorial design and classical design methods. In addition to introductory information on design, he covers issues including: regression, variance reduction techniques, optimization and many case studies. Kleijnen also wrote a text which addresses both the design and analysis of
simulation experiments (J. P. C. Kleijnen, 2008). Among the topics he investigates, is how simulation data may deviate from the standard experimental design assumptions and a section of designs for kriging. His coverage of kriging designs focuses on sequential designs, and so does not overlap with this work substantively.

Sanchez and Wan (2009) give a good introduction to the benefits and pitfalls of DOE in simulation studies. They begin by showing how simple efficient designs can break the curse of dimensionality. They outline an example where a brute force approach (requiring a petaflop processor more than 40 million years) can be outperformed by good design (requiring a high-end desktop a matter of days). They then outline many of the basics of design and some simple designs which can help reduce the computational burden. They give a handy table comparing the various benefits and limitations of many designs.

Carpenter (1993) examined the effect on response surface metamodel’s performance by a variety of designs. His results include problems with low and medium dimension, but are only used on deterministic functions. However, he gives a very comprehensive treatment and concludes with three rules for how many points should be required for a design, and what designs are best under certain circumstances.

Hurrion and Brigil (1999) examined how the choice of design and metamodeling technique effect performance on two manufacturing systems. They compared factorial and random sampling design for use in regression and neural net metamodels. They found that the ANN combined with randomized design seemed to be best. Our results will expand on
their work by using more designs, more problems, and situations without much data. However, we will not look at multiple metamodeling techniques like they did.

Alam et al. (2004) also employed ANNs. Their study only uses a single test-bed problem without exploration of the effects of noise and dimension. However, they do incorporate designs which utilize domain knowledge and shown doing so to be quite useful.

2.4 Methods

The methodology for this study was straightforward. There were six steps. First, we picked a set of designs to test. Second, we picked/created a set of test-bed problems. Third, we specified how the metamodels would be constructed. Fourth, we identified a set of metrics for evaluating the quality of various metamodels. Fifth, we designed and ran experiments testing the quality of the metamodels derived from each design on the testbed problems. Sixth, we interpreted those results.

Before we cover the five major steps (the sixth one is saved for later), an important minor step had to be addressed: what is the design space, \( \mathcal{X} \)? In practice, the analyst knows the design space to use because the various experimental controls are known, and those controls probably have upper and lower bounds which define the dimensions of \( \mathcal{X} \). However, we are testing methodologies over test problems and so the choice of design space is ours. This decision must be considered carefully to ensure the results are valuable. For simplicity
we decided to make the unit hypercube our design space. This design space is the most standard approach in literature and many real applications can be recast into it.

2.4.1 Choosing Designs

For Appendix A we researched and compiled a number of designs. We needed to reduce this list to a set of designs for testing. We used the following four principles to pick the designs to be tested:

- They are reasonably well-known;
- They cover a wide spectrum of experimental design theory;
- They can be coded up without much difficulty for any analyst; and
- They can work in any number of dimensions.

Appendix B provides a large number of figures which show each design we chose including representations in two and three dimensions to help illustrate them.

The first design chosen was uniform random sampling (RAND). In this design each component in $x_i$ are independent random numbers uniformly distributed between 0 and 1. This design is very simple to create, and handles augmentation (adding more points) quite well. It is included mainly to serve as a baseline, where if a design cannot perform better than it, the design in question is probably of little value.
The next design type picked was a factorial design. This design is probably the most frequently used design by analysts who have not considered good design. This design can be viewed as slicing the design space into equally sized sub-hypercubes and then placing a point in each sub-hypercube. It is standard practice to place those points such that the distance between points is maximized along each dimension separately. We called this design “edged” and placed points so they are maximally spread out (since it will place points on the faces/edges/vertices of $\mathcal{X}$). We also decided to include a design where the points were placed in the centers of their respective sub-hypercubes and called it “Centered.” One significant complication with these designs is the curse of dimensionality. As the dimension of a problem grows, the number of points required by this design grows exponentially. Since we are in situations where SM is necessary, we can assume that data is limited, and so an exponentially growing demand on points is impossible to meet. This problem has been addressed already by an experimental design in the form of the fractional factorial design. For our testing, we choose to do a full factorial design if possible, and then resort to creating a fractional factorial design if there are insufficient points. We then named these designs the full/fractional factorial design, edged (FFDE) and the full/fractional factorial design, centered (FFDC).

Next we turned to central composite design (CCD) for a design to test. This design takes some core design, normally a factorial design with two levels, and then adds $2d$ axial points and some number of replications of the center point. For deterministic situations, there is no need for multiple center points. However, in the case where there is stochastic noise
(true in most OR simulations) extra center point replications are valuable. For our number of replications at the center we choose to use half of the dimensionality ($d$) rounded down, subject to a minimum of three replications and a maximum of seven. The number of replications is formally given as $\max(\min\left(\left\lfloor \frac{d}{2}\right\rfloor, 7\right), 3)$. The question of where to place the axial points (how far out from the center) is one which has received some attention before. We choose to always set the axial points on the edges of the design space, a distance of .5 from the center point. This option was picked because we can only ensure the existence of a solution for our simulation problems inside the design space. This approach then left the question of how far out to let the core portion of the design to go. In the case of the faced central composite design (CCDF), it is designed to go all the way out to the edge of the design space (the fourth design in our set to be tested). In the case of the inscribed central composite design (CCDI) we needed to make a different choice. We decided to place the core points out as far from the center points as the axial points (seems to be the standard choice). This choice meant that the distance from the center point to the corners of the hypercube which contained the core design would be 0.5.

The formula for a point at a vertex of the hypercube centered around (.5, .5, ..., .5) is:

$$(.5 \pm a,.5 \pm a,\ldots,.5 \pm a), \text{ for all } 2^d \text{ combinations of } \pm$$
We specified that the distance from the center point (.5, .5, ..., .5) to these points should be .5. Hence we get:

\[
0.5 = \sqrt{\sum_{j=1}^{d} (0.5 - (0.5 \pm a))^2} \rightarrow 0.25 = \sum_{j=1}^{d} (\pm a)^2 \rightarrow 0.25 = d \cdot a^2 \rightarrow \sqrt{\frac{0.25}{d}} = a,
\]

which then means the coordinates of the corners of the hypercube depend on the dimension of the problem and are given by the following expression:

\[
\{x: x_i = 0.5 \pm \frac{1}{2\sqrt{d}} \quad i = 1, ..., d\}
\]

The sixth design we decided to include was a Number Theoretic Method (NTM) proposed by Fang (1994). Fang gives many different designs throughout the manuscript, so picking one was no easy task. We choose the “Good Point” method which is a derivative of the “Good Lattice Point” method. This method is best suited for work high dimension (>10) and is good for use by analysts (easy to augment designs and simple to implement).

The seventh design we choose to test was the Coffee House Design (CHD) given by Muller (2007). This design was picked because it is a reasonable implementation of optimal space filling without too great a computational burden. It also is more receptive than most space filling methods to augmentation later. While this method is it not as easy to implement when compared to other methods in this dissertation, it is still something an average analyst should be able to do without great effort.
Finally, we choose to include Latin hypercube designs in our testing. For our eighth design we choose a simple Latin hypercube sampling design (RLH). This design is popular in designing computer experiments due to its track record of success and simplicity. Next we decided to test a maxmin Latin hypercube design (MMLH) which, while keeping the Latin hypercube property, tries to maximize the minimum distance between any two points. The way we implemented this design was by utilizing the built-in Matlab function which creates this kind of design via a row exchange process. We choose to use a much higher number of iterations than the default because we found significant improvements in the criterion could be achieved beyond the default setting. Lastly, we choose for our tenth design a symmetric Latin hypercube design (SLH). This design is easy for an analyst to implement and gives another valuable idea in designs, symmetry. (J.-S. Park, 1994; Ye, Li, & Sudjianto, 2000)
The designs are summarized in Table 2. Matlab code for the creation of the designs is available from the author.

Table 2: Experimental Designs Summary

<table>
<thead>
<tr>
<th>Design</th>
<th>Abrev.</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Sampling</td>
<td>RAND</td>
<td>(K. T. Fang, Li, &amp; Sudjianto, 2005)</td>
</tr>
<tr>
<td>(Fractional) Factorial, Centered</td>
<td>FFDC</td>
<td>(Giesbrecht &amp; Gumpertz, 2011; Ryan, 2007)</td>
</tr>
<tr>
<td>(Fractional) Factorial, Edges</td>
<td>FFDE</td>
<td></td>
</tr>
<tr>
<td>Cent. Composite Design, Inscribed</td>
<td>CCDI</td>
<td></td>
</tr>
<tr>
<td>Central Composite Design, Faced</td>
<td>CCDF</td>
<td></td>
</tr>
<tr>
<td>Number Theoretic Methods</td>
<td>NTM</td>
<td>(K. Fang, 1994)</td>
</tr>
<tr>
<td>Coffee House Designs</td>
<td>CHD</td>
<td>(Müller, 2007)</td>
</tr>
<tr>
<td>Latin Hypercube Samplings</td>
<td>RLH</td>
<td>(K. T. Fang et al., 2005)</td>
</tr>
<tr>
<td>Maximin Latin Hypercube Design</td>
<td>MMLH</td>
<td>(K. T. Fang et al., 2005)</td>
</tr>
<tr>
<td>Symmetric Latin Hypercube Design</td>
<td>SLH</td>
<td>(K. T. Fang et al., 2005)</td>
</tr>
</tbody>
</table>

Before we could continue, we had to address the fact that some designs do not accommodate any number of points in any dimension. We desired flexibility when it came to running tests, so we needed some way to modify each method so that it could create designs of any size for any dimension (subject to some minimum number of points). Some of the designs already allowed for this feature. Specifically RAND, NTM, CHD, RLH, MMLH, and SLH all can be designed in an arbitrary number of dimensions with an arbitrary number of points. So that left how to modify the (fractional) factorial and central composite designs. In our modifications, we wanted to preserve the intent of these designs while still being something an average analyst could understand and employ. With those principles in mind,
we decided to make each design as large as possible, without exceeding the specified size \((n)\), and then add the remaining points via whatever could best approximate the motivation of each method.

For the (fractional) factorial design, we broke the problem into three cases. Those cases were based on the relative numbers of points in the design and the number of dimensions for the design. The three cases were: \(n < 2^d\), \(2^d \leq n < 3^d\), and \(n \geq 3^d\).

First, consider the case where \(n < 2^d\). In this case we do not have sufficient points to make even a full two-level design. Therefore, we must make a fractional factorial design. We will make the fractional factorial design with highest possible resolution, without exceeding \(n\). In the case of leftover points, we will randomly pick corners of the hypercube not included in our design already.

Second, consider the case where \(2^d \leq n < 3^d\). In this case we do have sufficient points to make a full two-level design. So we start by making a full two-level design. In the case of leftover points \((2^d < n)\), we will add the points which would make up a three-level design but are not in a two-level design.

Third, consider the case where \(n \geq 3^d\). In this case we can make a full factorial design with at least 3 levels. This case means we either have abundant data acquisition capabilities or are dealing with a low-dimension problem. We chose to make the largest possible balanced factorial design. There is no way to slip in leftover points into the grid.
without breaking the lattice structure. So we add in points sampled randomly throughout the space (RAND design).

The central composite designs handled the creation of the core design and leftover points in the same way as the (fractional) factorial designs. However, we set aside $2d + n_{cent}$ points (the number of center replications and axial points) before creating the core of the design. If one of the points were to be duplicated in the design (both in the core and center/axial), we made sure that a different point was added instead.

One might be curious how the SLH designs can accommodate an odd number of points (it is trivially clear that an even number of points would be allowed). The addition of the center point, (.5, ..., .5), gives us an odd number of points and does not violate the symmetry. This approach may be a slight violation of the Latin hypercube design requirements, but it seems to be the best choice available.

2.4.2 Choosing Testbed Problems

Now that the designs to be tested were determined, we turned our attention to the problem of creating a test-bed of simulation problems. We wanted our test-bed to include problems with a variety of features so that we could distinguish not only the best overall design, but what problem features cause different designs to perform better.
There were five properties which would be required of all problems. First was that the problem could be run without massive computational burden. While in the real world, problems which benefit from SM regularly require large amounts of computation, for our testing of methods, we did not have years to spend on running experiments. The second property was that the problems could work in an arbitrary number of dimensions. This condition was needed so that we could study the effects of varying the dimensionality of problems on our results. The third property we deemed necessary was that each problem could have a variable amount of noise added to the data. In particular, we chose to quantify the ratio of signal to noise. Signal to noise is important for OR simulations because they normally have some estimable amount of noise at each point.

The fourth property was that each problem would be unimodal. This property was necessary because we knew that we would be working in high dimension spaces and working with multimodal problems would become a futile effort. Without some structure to exploit, there is little hope to produce a high-fidelity metamodel of a high-dimensional problem with multimodality; the curse of dimensionality is too strong. The fifth property was that each problem would have some random elements which would allow for multiple distinct realizations. The realizations would all still have a consistent form and set of features (discussed next); but the variation from realization to realization would give us more robust results and allay fears that a design might be unfairly biased positively or negatively in evaluation.
As we added problems to the test-bed and began to run experiments, we noticed some features of certain problems that seemed to impair or augment the performance of certain designs. The first feature we found to be critical to the performance of designs was the behavior of the problem at the edges of the design space. In particular we found that when a problem had explosive growth on the edges of the design space (as it is present in a queuing model) it wreaked havoc with some metamodels’ predictions. This observation led us to realize that the edge behavior was an important feature to consider in problem selection.

The second feature we found necessary to control was the location of the mode (optimal value). In particular, we needed to distinguish between problems with the optimal values at a vertex of the design space versus those with the optimal value in the interior.

To address the combinations of factors listed above, we picked test-bed problems from the literature and then created problems which would fill in the gaps. In Appendix C: TESTBED PROBLEM FORMULATIONS AND ILLUSTRATIONS we give both formal definitions of each problem and illustrations.

The first problem in the test-bed was also the simplest, named “Slope.” Slope (SL) is simply a linear function in each variable. It is totally separable, has no curvature, and it has its optimal value at a vertex. The second problem we included was “Hyper-J.” Hyper-J (HJ) is again separable, but this time instead of being a linear function, it is curved with explosive growth on one edge of \( \mathcal{X} \) for each dimension. It also has its optimal value at a vertex. It draws its name from what its one-dimensional cross sections look like, Js. The third problem added was called “News Stand.” This problem is based on a “newspaper boy” problem, but
extended into many dimensions with product substitution. The news stand problem (NS) is smooth and quadratic in shape without any explosive growth on edges. The fourth, and final, problem included was named “Queuing Network.” Queuing Network (QN) is the problem of finding how to set service rates at different nodes in a queuing network. This problem is quadratic looking near the optimum, looks linear in some other areas, and has explosive growth near one edge of $\mathcal{X}$ for each dimension (resultant from insufficient service at a node and so an unstable system)

2.4.3 Metamodeling Construction

Now that we had our designs and test-bed problems picked, we needed to determine the specifics of the metamodeling process. As was stated in the literature review, we chose universal kriging because of its robustness, recent popularity, and uniformly excellent track record. Since this work is focusing on the most common usage of SM and creating a preliminary metamodel is generally required. So when we had to make various choices in the metamodeling process, we chose to do what would probably be done in preliminary stages.

There were three key questions to be answered in the metamodeling process: the form of the regression function (since we are using UK), the correlation function's form, and the method for estimating the correlation function's parameters. First, for the regression function, we chose to use a linear model. Generally, linear models will not overfit, which can be a concern. (Babyak, 2004; Hawkins, 2004) Further, for linear models the number of data points
needed is on the order of the dimension of the problem. Pragmatically, in situation requiring SM we likely do not have data points on the order of the dimension squared or more, when the dimensionality of the problem is high or even medium sized. Hence using linear models is reasonable. The question of using separable higher order models (those with second order terms but no interactions) will be left open for future work.

Second, we decided to use an anisotropic exponential correlation function with nugget effect \((0 \leq \theta_0 \leq 1)\) for all our metamodels. This is shown below as Equation 13:

\[
C(x, x') = \begin{cases} 
\theta_0 \prod_{i=1}^{d} e^{-\theta_i h_i} & \text{if } x \neq x' \\
\prod_{i=1}^{d} e^{-\theta_i h_i} & \text{if } x = x'
\end{cases}
\]

where \(h_i = |x_i - x_i'|\).  

Equation 13

This choice of correlation function further protects us from overfitting since it is a simple model. This is critical when dealing with the little data available in a SM context. Ultimately we do not know the perfect correlation function form to use for any problem. For a particular problem, an analyst should look at their data carefully to select the most appropriate one (Michael L Stein, 1999). We however do not have that luxury since we are doing our fitting in batch and so as to not bias the results with human decision making.

Third, we decided to estimate the correlation function's parameters via maximum likelihood estimation. While this method is much more expensive computationally than regression estimation of the parameter values, it the choice frequently made in statistics (Booker et al., 1999; Martin & Simpson, 2005) or recommended in literature (Simpson et al.,
Lastly, since we are in a situation where the limiting factor is data acquisition (runs taking hours or more), the addition of a matter of seconds (what MLE requires) is negligible.

2.4.4 Metamodel Evaluation

Now that we had designs picked, problems created, and metamodelling methodology specified, we needed decide how to evaluate the quality of each metamodel. In particular, we needed some metrics for comparing the metamodels and a testing procedure to follow through the testing. It is important to note that since this is a methodological study, we chose to use theoretical metrics which needed knowledge of the true underlying functions from which the data was created.

In literature, the most common measure of quality is some form of average prediction error. This average error measurement normally takes the form of mean squared, mean absolute, or root mean squared error of predictions. We settled on using mean absolute error of predictions (MAE). Another metric often found in literature is some form of maximum error in prediction. We decided that by including the maximum error of prediction (ME) and the MAE of prediction we would get both a sense of how a metamodel did at predicting on average as well as in the worst case, both of which are important.

Next we chose to create (what we believe to be) a novel measure in evaluating the metamodels. The first two measures look at the fidelity of the surface created in simple terms of deviation from the true value. Since we are so frequently interested in improving
optimization via a metamodel surrogate, we wanted to add a measure which explicitly accounted for that need. In Figure 4 we show a function (black solid line) and three possible surrogate metamodels for it (red with short dashes, blue with long dashes, and green with dot dash). Note: for clarity and consistency we will treat all problems as ones where we desire maximization of the response for optimization purposes.
We see that the red with short dashes function is clearly the best surrogate from perspective of both MAE and ME, followed by the blue with long dashes and the green with dot dash. When it comes to optimization we are most interested in a surrogate which identifies the optimal solution $x^*$. If we were to try and use the red surrogate for optimization, we would find it misleading. In fact, it would be the worst of the three potential surrogates. The surrogate in blue with long dashes would be our second choice in both MAE and ME, however it too would be less useful in optimization than the green with dot dash.

What we have identified with this example is the need for a surrogate used for optimization
to discriminate among possibilities in the design space. To measure this ability to discriminate, we created a measure we named the “Alpha Criterion.”

The Alpha Criterion (AC) seeks to measure and quantify the ability of a metamodel to identify the most promising areas for optimization. Specifically, the AC measures a metamodel’s success rate in identifying the top alpha proportion of a data set. The steps to get an AC value are detailed below:

1. An alpha level, $\alpha^*$, must be specified. ($0 < \alpha^* < 1$)

2. A set of test points must be predicted by a metamodel and the true function. Let the number of test points be denoted $N_{TP}$

3. Rank the test points by largest response to smallest based on the true response data and identify the top $\lceil N_{TP} \alpha^* \rceil$ of them. Call that set $T$

4. Rank the test points by largest response to smallest based on the metamodel’s predictions and identify the top $\lceil N_{TP} \alpha^* \rceil$ of them. Call that set $P$.

5. The alpha criteria value for given alpha level $\alpha^*$, is found via:

$$|T \cap P|/|T|$$
Figure 5 gives an example of this process. In this case the alpha level is .05 and there are 100 test points.

In Figure 5 we see that there are five (100*.05=5) points which are marked with red crosses. These points are the points which are the true top five points in the function. We also see five black squares, these are the top five points as predicted by the metamodel. We see that in three of the cases the metamodel properly identifies a top five points. In the other two
cases it misses on its prediction. This means that at the alpha .05 level we would say this metamodel performs at 60%.

To utilize the AC, we found it necessary to use it at multiple levels. Originally, we wanted to use it at a low alpha level (1%-5%), a medium level (10%-20%) and a high level (25%-50%). Using it at these three levels would give us a better sense of how the metamodel performs at picking the best areas for optimization, the areas which deserve further study, and the areas which can be safely screened out of further study. So we decided to average the alpha performance at the 5%, 10%, 20%, and 40% levels into a single value. However, we found that the AC then did not discriminate between the quality of performances of various metamodels. We found that by adding additional alpha levels we were able to get more discrimination for fewer replications without significant computational increase (essentially adding more power in our testing). We also found that doing so had no negative impact in which metamodels were found to be superior. Even with this change the power of the AC to discriminate was still weaker than the other two metrics, but we found it to be useful so we kept it. We settled on using alpha levels of 5%, 10%, 15%, …,40%.

Now we had three metrics for comparing metamodel performance: mean absolute error of prediction, maximum error of prediction, and alpha criterion. However, we found that these metrics, specifically MAE and ME are quite vulnerable to outliers. These outliers can then skew how we perceive the performance of each design’s metamodels. This observations led us to rank each metamodel on each metric for each replication. By analyzing these rankings, we protected ourselves from outliers. Further, this modification meant that we
did not need to worry about the relative size of the errors in prediction between different test-bed problems or do any scaling to account for those differences. By using rankings we risk the possibility that we exaggerate certain differences between metamodels which are actually very similar in performance. However, this risk is outweighed by improvements in resistance to outliers and removal for need for scaling.

2.4.5 Experimental Procedure

Each basic replication for testing followed this process:

- Create a realization of a problem;
- Obtain data at the points specified by each design on the problem realization;
- Use the data from each design to create a metamodel for each design;
- Pick a set of test points (randomly) and get their values on the problem realization;
- Use each metamodel to predict at the set of test points;
- Evaluate the quality of each metamodel (Comparing steps 4 and 5); and
- Rank the performances.
Now there were still five questions to be examined:

- How many dimensions will each problem be tested at?
- How many data points will be used in each design?
- How many data points will each metamodel try to predict at?
- What levels of signal to noise ratio will we use?
- How any replications will we do for each problem-dimension-noise 3-tuple?

First, we addressed the problem of how many dimensions we would test. We wanted to test at wide variety of dimension levels, particularly the medium and higher ones which are not commonly present in literature. The initial range we picked was 2 through 30. This upper limit was imposed because the generator we used for the fractional factorial designs only could accommodate up to 32 dimensions. Work into higher dimensions has been left for future work. Further, Kleijnen (1998) gives a convincing argument for screening at higher levels of dimensionality before engaging in detailed metamodeling as in this work.

Next, we had to decide how many points to include in the designs. Our goal is for our results to be of value in situations with limited data. So the natural question is how limited is “limited data?” The answer is that it varies based on how high dimensional the problem is and how difficult the problem is to model. As the number of points and modes increases, it requires a greater number of points to have an equivalent amount information. To illustrate this issue, Figure 6 shows the relationship between dimensionality and the distance between points. The solid black line in Figure 6 shows the average distance between 100 randomly placed points on the unit hypercube. The dashed red line shows the average distance between
a point and its farthest neighbor in those 100 points. Lastly, the short dashed blue line shows the average distance between a point and its nearest neighbor.

![Figure 6: Relationship between Dimension and Distance](image)

We can see from these plots that these distances grow on the order of $\sqrt{d}$. This growth continues without bound as dimensionality increases. Furthermore, while 100 points in two dimensions may seem stacked close together, when we have as few as six dimensions, the average distance between any two points is one unit. Further, when the dimension grows to about 15 the distance between an average point and its nearest neighbor is one unit.
The second component in deciding how limited is “limited data” is the problem characteristics. A large amount of curvature can make it difficult to build a high fidelity surrogate. This characteristic becomes particularly troubling when the dimensionality of a problem is high and so the difficulties multiply.

Ultimately, we choose to set a consistent baseline: we decided to use 100 points for all tests. This was in large part motivated by our choice of maximum dimension (the third question). We need to have at least slightly more than $3d$ points to be able to build the CCD designs. Also our modeling had just over $2d$ degrees of freedom and so we need more points than that such that we would not have an underdetermined model.

Third, we decided on how many test points to use for each problem realization. In the end we settled on 100 test points. To find this number we ran successive tests with greater and greater numbers of test points. We then waited until the results converged to our satisfaction and then used the fewest number required. We found that using more than 100 seemed to not change the results or add significantly greater stability. On the other hand, using fewer than 100 seemed to be insufficient to give confidence with a small number of replications.

The fourth question was what levels of signal to noise to include. This choice was a very difficult decision to make. It is well known that simulation models have different variances at different combinations of inputs. In statistical parlance, this is often called heteroskedasticity, while in spatial statistics it is often called non-stationarity. This variability is problematic because the standard approach in spatial statistics for non-stationarity is for
the analyst to intervene and “look” at the data and the various measures of it. The analyst then cobbles together a plan for how to proceed with the analysis. This approach is impossible for a methodological study (like this) because we would likely introduce bias based analyst skill in doing the analysis. Further, there would be difficulties due to the time required for an analyst to work on each data set as is proper.

So we decided we needed to use something simpler. First, we estimated the variance of a problem over the entire design space. Next, we added normally distributed noise with mean zero and variance based on our first step and then signal to noise ratio to be studied for that problem instance. This process is detailed in the end of Appendix D.

We added this noise to the test-bed function values which the metamodels receive. We did not add noise to the points which the metamodels would predict because it is useful for a metamodel to be able to filter through the error present in data and still to give good predictions.

Lastly, we need to decide on how many replications we would run for each 3-tuple (which problem, how many dimensions, and what level of noise). This process was performed by running a pilot set of replications (10) and then augmenting that set with further replications until we felt the results had converged. In the end we did 120 of these replications.
2.5 Results

The original intent of our approach was to find a design that is the “best default choice.” However this is not a simple matter. First, there is the issue that one design may superior on one metric but inferior on another. Second, there may be various factors such as problem shape, dimensionality, and/or noisiness which affect those results. We needed to find a method for resolving those conflicts. We decided to look for dominance relationships between various designs. We hoped that we could (and were able to) eliminate enough designs via dominance that we could make compelling conclusions about which design are best under what conditions. In end, our results are instructive, but not comprehensive (leaving room for future work).

We decided one design dominated another if it was superior on all three metrics. This brings up a new question: “what does ‘superior’ mean?” We first settled on a reasonably strict definition which mimicked stochastic dominance. We call this ranking dominance.

Design A ranking dominates (RD) design B if for all ranking levels design A is at least as frequently ranked at that level or better than design B. For example, Design A is said to ranking dominate design B for a given metric, if:

- A is ranked as the best design for the given metric at least as frequently as B.
- A is ranked best or second best at least as frequently as B is rank best or second best
- This continues (A being better more frequently than B) for all possible ranking levels.
We then amended this definition to address a concern. By the strictest adherence to this definition, two designs which had the same distribution of rankings could both dominate each another. To address this problem we added that there must exist at least one ranking level for which A strictly is better than B.

From this concept of RD, we can explain dominance between designs. If one design ranking dominates another on all three metrics we can say without reservation it is superior. We termed this strict dominance (SD). As desired, we now had method to start reducing the field of designs. If a design is strictly dominated over some subset of the data, then it can be disregarded for use over the subset on which it is dominated. However, we later found it useful to also create a weaker form of dominance which we termed weak dominance.

For design A to weakly dominate (WD) design B, we require two components. First, A must have a better average (mean) ranking than B on all three metrics. Second, design A must ranking dominate design B when the rankings for all three metrics are combined. While weak dominance is not airtight, it is strong enough that we are comfortable using it to exclude designs. We will take care to distinguish between the use of strong dominance and weak dominance throughout the results section.

If a design is dominated on a subset of the data we are interested in, we will then feel free to exclude it from the analysis on that subset. This exclusion is safe since there exists at least one design which is superior on all three metrics. Hence there is no reason to ever use a dominated design. We then can remove that design from the data and rerun the rankings and analysis. This process of ranking, identifying dominance, and excluding can then be repeated.
until only some final subset of designs remain (possibly just one). It may not be evident initially how repetition of this process would be useful, but similarly to how dominance and exclusion in game theory merits repetition, the same applies here.

2.5.1 Problem 1: Slope

This problem’s simplicity makes it easy to model regardless of what design was used. Despite each design’s success in modeling, one design was head and shoulders above the rest, the FFDE. The FFDE strictly dominates each other design. The fact that it dominates overall is due to its strength as the problem becomes more difficult (higher dimensionality and/or more noise).

If we break the dimensionality and noisiness spectrums into thirds and the take the resultant subset of the data we see a bit of variation. Table 3 shows the breakdowns by dimensionality and noise. The column below Dimensionality (the 2nd one) shows the non-dominated designs for a given noise level (shown in the first column) over all subsets of dimensionality. Similarly, the row starting with Noise Level (the second one) gives the non-dominated designs for a given level of dimensionality (in the top row) over the full set of noise levels. In the lower right 3 by 3 subsection of the table, we see the non-dominated designs for the given noise level and dimensionality. If a design is present in a cell then it mean that was not SD by any deign, and therefore is at least a viable candidate for that situation.
Table 3: Effects of Dimensionality and Noise on FFDE's Dominance for the Slope Problem

<table>
<thead>
<tr>
<th>Noise Level</th>
<th>Dimensionality 2:2:10</th>
<th>Dimensionality 12:2:20</th>
<th>Dimensionality 22:2:30</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:.1:.6</td>
<td>3</td>
<td>3, 4, 5, 6, 8, 9</td>
<td>3</td>
</tr>
<tr>
<td>.7:.1:1.3</td>
<td>3</td>
<td>3, 5</td>
<td>3</td>
</tr>
<tr>
<td>1.4:.1:2.0</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>


These results tell us that if an analyst believes their problem to have its optimal value at a vertex and the problem to have minimal (or no) curvature, they should utilize a design which put the maximum effort at the vertices. Further these vertices should be selected via an FFD if there are insufficient points to cover each vertex. FFDE is without reservation the best design for this problem.

2.5.2 Problem 2: Hyper-J

This problem is not an easy one to model. Its combination of difficult-to-model edge behavior and having its optimal value at a vertex, makes it impossible for any single design to emerge above the others. Designs which place lots of their points near the edges of the design space (FFDE, CCDF, and to a lesser extent CHD) are able to identify the optimal solutions well (AC), but are poor on overall fidelity (ME and MAE). On the other hand, designs which keep points away from the edges of the design space (LH and NTM) perform poorly on AC and well both ME and MAE. These results are shown in Table 4.
Table 4: Average Ranks for Each Metric on Hyper-J Problem

<table>
<thead>
<tr>
<th>Design</th>
<th>Avg ME Rank</th>
<th>Avg MAE Rank</th>
<th>Avg AC Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAND</td>
<td>3.5911</td>
<td>3.3292</td>
<td>5.9136</td>
</tr>
<tr>
<td>FFDC</td>
<td>6.3772</td>
<td>5.9235</td>
<td>6.5159</td>
</tr>
<tr>
<td>FFDE</td>
<td>9.6843</td>
<td>9.8453</td>
<td>4.1090</td>
</tr>
<tr>
<td>CCDI</td>
<td>6.8567</td>
<td>7.1042</td>
<td>5.6002</td>
</tr>
<tr>
<td>CCDF</td>
<td>8.7739</td>
<td>8.9417</td>
<td>3.9053</td>
</tr>
<tr>
<td>NTM</td>
<td>3.5474</td>
<td>2.9434</td>
<td>5.8363</td>
</tr>
<tr>
<td>CHD</td>
<td>5.6929</td>
<td>7.1466</td>
<td>5.5419</td>
</tr>
<tr>
<td>RLH</td>
<td>3.4241</td>
<td>3.1414</td>
<td>5.8160</td>
</tr>
<tr>
<td>MMLH</td>
<td>3.4157</td>
<td>3.1460</td>
<td>5.8049</td>
</tr>
<tr>
<td>SLH</td>
<td>3.6367</td>
<td>3.4788</td>
<td>5.9569</td>
</tr>
</tbody>
</table>

Next we will focus on each specific metric in turn. On the criteria of ME, we found that the MMLH is the best choice. It ranking dominates all the other designs on that criteria. This is not a surprise, as it did have the best average ranking on the ME criteria in Table 4.

After the ME, we looked at the MAE. This metric pointed to the NTM as superior to all others (RD). Again, this is not surprising since the NTM had the best average rank on the MAE criteria. This fact combined with the previous findings on ME leads us to begin forming a useful conclusion. The conclusion is designs which spread their points out uniformly over the design space do the best jobs at creating high-fidelity metamodels for this problem. Further confirming that point is the fact that all the LH models perform well on both ME and MAE.

Finally, we look at the AC. The CCDF is the best design for this criterion (RD). Again, this result should be anticipated since CCDF had the lowest average ranking on AC.
The second best design on AC was FFDE, it was competitive with the CCDF, and it was the only design which took two iterations of dominance relations to be dominated. Both these designs perform well on AC and abysmally on ME and MAE. This result also provides evidence for the conclusion from the previous paragraph, adding to it that designs which put many points near the edges of the design space perform better on AC. Later, we will generalize and formalize this finding.

2.5.3 Problem 3: Newsstand

This problem is not difficult to model, as it is essentially quadratic everywhere. However, by moving its optimal value into the interior of the design space, we would expect the designs with fewer points on the edges of the design space to perform better, particularly on the AC. What we find does follow those results. While only five of the designs can be eliminated via SD (RAND, FFDC, FFDE, CCDI, and NTM), using WD only two are left: CHD and SLH.

Looking again at the various criteria, we find the CHD to ranking dominate all the other designs (including SLH) on both ME and MAE. On the other hand we find SLH to be superior on AC (RD). This split is what we would expect, given no one design dominates. What this tells us is that the way to create the metamodel with the highest fidelity is to distribute points evenly over $\mathcal{X}$, but away from the edges of $\mathcal{X}$. This should give a good mix of information about the curvature and the local variation. On the other hand, since the CHD
was best for the AC, we can conclude that utilizing points which are very spread out, and on the edges of $\mathcal{X}$, garners the most information about the location of the optimum value. This is further confirmed by the fact that the CCDF and MMLH had the second and third best average ranks on AC respectively.

2.5.4 Problem 4: Queueing Network

This problem is very important for OR, as it forms the basis for a tremendous number of simulation applications. We find that the NTM, RLH, and MMLH strictly dominate all the other designs, but not each other. Considering weak domination does not change things; all three of those designs remain viable.

The NTM RD on both MAE and AC. Being best on those two criteria means it is likely the best design for most applications. It is also an interesting combination. Normally, we would expect that if a design was going to RD on two criteria (as opposed to zero, one, or three) the two criteria would be ME and MAE. The reason NTM does not RD on ME is that it keeps its data points away from the edge of the design space more than RLH and MMLH. This means that when asked to predict at a point on the edge of the design space there is a greater chance it will not have a data point nearby and have a truly awful prediction. This hurts its performance on ME.
2.5.5 Summary of Results by Problem

A succinct way of thinking of the sets of designs which are not dominated is an efficient frontier. Consider a design’s performance as a point in \( \mathbb{R}^3 \) with one dimension for each metric. We then would see these non-dominated designs as forming some frontier which the other designs are contained inside. Looking at the RD design on each metric, we would then have design furthest out on each dimension, myopically ignoring the other metrics. In Table 5 we summarize our findings on all four problems.

Table 5: Dominance by Problem

<table>
<thead>
<tr>
<th>Problem</th>
<th>Designs Remaining after</th>
<th>RD Best Choice by Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Strict Dom</td>
<td>Weak Dom</td>
</tr>
<tr>
<td>Slope</td>
<td>FFDE</td>
<td>FFDE</td>
</tr>
<tr>
<td>Hyper J</td>
<td>All but RAND and FFDC</td>
<td>FFDE, CCDI, CCDF, NTM, CHD, RLH, MMLH</td>
</tr>
<tr>
<td>Newsstand</td>
<td>CCDF, CHD, RLH, MMLH, SLH</td>
<td>CHD, SLH</td>
</tr>
<tr>
<td>Queueing Network</td>
<td>NTM, RLH, MMLH</td>
<td>NTM, RLH, MMLH</td>
</tr>
</tbody>
</table>
2.5.6 Results by Noise and Dimensionality

Next we wanted to observe the effects of dimensionality and noisiness on our results. We would have liked this analysis to proceed using information from all of the problems in our test-bed. However, when we try to use our previous tools (RD, SD, and WD) we run into the same problem which drove us to break things down by problem previously, namely the lack of conclusive results. In Table 6 we show the results after removing weakly dominated designs for various subsets of the data (a legend is at the bottom of the table).

Table 6: Weak Dominance by Dimensionality and Noise

<table>
<thead>
<tr>
<th>Noise Level</th>
<th>Dimensionality</th>
<th>2:2:10</th>
<th>12:2:20</th>
<th>22:2:30</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:.1:.6</td>
<td>10</td>
<td>6, 8, 9, 10</td>
<td>6</td>
<td>3, 7, 8, 9</td>
</tr>
<tr>
<td>.7:1:1.3</td>
<td>3, 6, 7, 8, 9</td>
<td>6, 8, 9, 10</td>
<td>6</td>
<td>3, 7, 8</td>
</tr>
<tr>
<td>1.4:1:2.0</td>
<td>3, 6, 7</td>
<td>6, 8, 9</td>
<td>3, 5, 6, 7</td>
<td>3, 7, 9</td>
</tr>
</tbody>
</table>


The column below Dimensionality (the 2nd one) shows the non-dominated designs for a given noise level (shown in the first column) over all subsets of dimensionality. Similarly, the row starting with Noise Level (the second one) gives the non-dominated designs for a given level of dimensionality (in the top row) over the full set of noise levels. In the lower right 3 by 3 subsection of the table, we see the non-dominated designs for the given noise level and dimensionality.
From inspection of this table we see a few patterns emerging. For the easier problems (those with low dimensionality and noise, in the top left) we see the NTM and SLH design show up as non-dominated. On the other hand, difficult problems (high dimensionality and noise) are best served by FFDE and CHD. From this we can speculate that on the easier problems, designs which put more points on the interior of the design space do better, and for harder problems, putting the points on the edges seems best.

The intuitive explanation is that when problems are easier, we have more points available to explore over the design space, particularly including the interior. This excess of points then means that incorporating extra features like symmetry and uniform design become reasonable and we can get their added benefits without much cost to coverage. On the other hand, when problems are difficult we are best off attempting to extrapolate behavior in the interior by generalizing from the edges. The usefulness of this extrapolation may be due to the specific test-bed problems chosen. For the test-bed problems, using the vertex points to extrapolate a first order approximation (the heart of our UK method) is not a horrible approximation. The major exception to this is the queueing network problem, which can have its interior very poorly approximated by points from the edges of the design space.

2.5.7 Results by Problem Characteristics

The next topic we investigated was which characteristics of problems affect the performance of designs. We found this topic to be critical in deciding what design to use. We
ultimately found two interesting relationships which looked at the interplay between problem characteristics, design features, and the metric being used. For both observations the relationship of interest is the one between:

1. How much of the data is placed on the edges of the design space versus the interior for a given design; and

2. The edge behavior of the given problem, specifically how it differs from that of the interior space.

The two observations are distinct because the three metrics break into two groups. First, we note that ME and MAE behave in one way with regards to factors one and two. Second, AC varies in a distinct way with factors one and two.

To study this we chose to look at 3 pairs of designs which use the same basic principles in design, except they differ on how much of their data they place on the edges of the design space. The three pairs are: FFDC and FFDE, CCDI and CCDF, and NTM and CHD. In each of those pairs the first design places fewer points on the edges, whereas the latter places more points on the edges of the design space. To look at the effects of edge behavior in problems we will consider which of the designs in each pair performs better on each metric for each problem.
Table 7: Paired Design Performance on ME and MAE

<table>
<thead>
<tr>
<th>Problem</th>
<th>ME</th>
<th>FFDC</th>
<th>FFDE</th>
<th>CCDI</th>
<th>CCDF</th>
<th>NTM</th>
<th>CHD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (SL)</td>
<td>.3277</td>
<td>.6723</td>
<td>.1584</td>
<td>.8416</td>
<td>.4414</td>
<td>.5586</td>
<td></td>
</tr>
<tr>
<td>2 (HJ)</td>
<td>.9490</td>
<td>.0510</td>
<td>.9567</td>
<td>.0433</td>
<td>.7761</td>
<td>.2239</td>
<td></td>
</tr>
<tr>
<td>3 (NS)</td>
<td>.8810</td>
<td>.1190</td>
<td>.2746</td>
<td>.7254</td>
<td>.4719</td>
<td>.5281</td>
<td></td>
</tr>
<tr>
<td>4 (QN)</td>
<td>.9699</td>
<td>.0301</td>
<td>.9215</td>
<td>.0785</td>
<td>.7714</td>
<td>.2286</td>
<td></td>
</tr>
<tr>
<td>MAE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 (SL)</td>
<td>.1881</td>
<td>.8119</td>
<td>.0843</td>
<td>.9157</td>
<td>.3872</td>
<td>.6128</td>
<td></td>
</tr>
<tr>
<td>2 (HJ)</td>
<td>.9889</td>
<td>.0111</td>
<td>.9728</td>
<td>.0272</td>
<td>.9592</td>
<td>.0408</td>
<td></td>
</tr>
<tr>
<td>3 (NS)</td>
<td>.9120</td>
<td>.0880</td>
<td>.5624</td>
<td>.4376</td>
<td>.6870</td>
<td>.3130</td>
<td></td>
</tr>
<tr>
<td>4 (QN)</td>
<td>.9988</td>
<td>.0012</td>
<td>.9594</td>
<td>.0406</td>
<td>.9183</td>
<td>.0817</td>
<td></td>
</tr>
</tbody>
</table>

Table 7 shows the proportions of time each design is superior to its partner on each problem and metric. In problems one (Slope) and three (Newsstand) the edges behavior is easy to model, and not unlike the behavior through the rest of the design space. In problems two (Hyper J) and four (Queueing Network) edge behavior deviates strongly from the interior behavior. To control for where the optimum value is (interior of $\mathcal{X}$ versus vertex of $\mathcal{X}$), we focus our comparisons on the performance on problems 1 to 2 and 3 to 4. We see in all 12 cases the “edgy” design (FFDE, CCDF, or CHD) performs worse on problems two and four relative to their performances on problems one and three (often drastically). This is not to say in every case for problems one and three the “edgy” design is better, or that in every case for problems two and four the “edgy” design is worse. Only that the “edgy” designs’ performance improves as the behavior of the problems on the edges becomes more like that in the interior.
The second observation considers the location of the optimal value (at a vertex or in the interior) and the usefulness of the metamodel for optimization. Since this observation refers to optimization it shows up in the AC performance over the pairs of designs.

<table>
<thead>
<tr>
<th>Problem</th>
<th>FFDC</th>
<th>FFDE</th>
<th>CCDI</th>
<th>CCDF</th>
<th>NTM</th>
<th>CHD</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC 1 (SL)</td>
<td>.3810</td>
<td><strong>.6190</strong></td>
<td>.2642</td>
<td><strong>.7358</strong></td>
<td>.4634</td>
<td><strong>.5366</strong></td>
</tr>
<tr>
<td>AC 3 (NS)</td>
<td>.4351</td>
<td><strong>.5649</strong></td>
<td>.3407</td>
<td><strong>.6593</strong></td>
<td>.4109</td>
<td><strong>.5891</strong></td>
</tr>
<tr>
<td>AC 2 (HJ)</td>
<td>.2700</td>
<td><strong>.7300</strong></td>
<td>.3227</td>
<td><strong>.6773</strong></td>
<td>.4667</td>
<td><strong>.5333</strong></td>
</tr>
<tr>
<td>AC 4 (QN)</td>
<td><strong>.8238</strong></td>
<td>.1762</td>
<td><strong>.6655</strong></td>
<td>.3345</td>
<td><strong>.7230</strong></td>
<td>.2770</td>
</tr>
</tbody>
</table>

In Table 8 we show the proportion of time that each design is superior in its pair on AC. Here the comparisons focus on problems one (Slope) to three (Newsstand) and two (Hyper-J) to four (Queueing Network) [note the swap of rows two and three from Table 7]. Making the comparison this way controls for the edge behavior in the problems (previously covered). In five of the six comparisons the more “edgy” design does better on AC when the optimum is at a vertex *relative* to its performance with the optimum in the interior of $\mathcal{X}$. This second observation demonstrates that to get a metamodel most useful for optimization, putting points near the optimal value is helpful.
2.6 Conclusions and Extensions

Picking the right design for a given problem can provide important benefits in the quality of the metamodel created. However, intelligently picking a design is not an easy task. In this work we have covered what factors should be considered by an analyst facing this choice. In this section we will discuss and summarize these results and then give directions for future extensions.

The overarching conclusion from this work is the value for analysts when they estimate how their problem is likely to behave and then clarify what they intend to use the metamodel for. We showed in the results section that the particular problem being studied vastly changes which designs are optimal. So, estimating the features of your problem (most notably the edge behavior and where you expect the optimum) is critical. This advice is not particularly innovative, but it is often overlooked by analysts in a rush to start using their models.

The second element we found important, but is often overlooked in practice and literature, is the intended use for the metamodel. Many times people simply state they want a "good," "representative," or "high fidelity" metamodel without considering what that means for their purposes. As we demonstrated in the methods section, there can be many competing metrics for "good."

The consideration of metric and problem characteristics cannot be taken independently. Towards the end of the results section, we explored the complex relationship between how much data a design places on the edge of the design space, the edge behavior of
a problem, the location of the optimal value, and the metric being favored. Specifically, we showed that on problems with difficult-to-model edge behavior, designs favoring edge placement of data do poorly on overall fidelity (ME and MAE). However, when in doubt it seems safest to favor more placement towards the center. On the other hand, when we are focused on optimization (AC) recommends putting data where we think the optimum is (interior vs. edge of \( \mathcal{X} \)), but when in doubt, favor edge placement.

There are many interesting extensions to this work. The more obvious extensions are to include are more problems and higher dimension problems. There can never be a complete set of test-bed problems; however that should not dissuade one from trying to grow the test-bed. Expanding the test-bed is a way that the results from this work could be made more robust and comprehensive. The first extension which we would pursue is looking at the effects of multimodality on metamodeling.

There are some interesting extensions regarding the designs tested as well. A simple one would be looking for how many replications at the center point in CCDs. However, we do not expect this extension to be valuable to follow since the CCDs were not found to be among the most useful. Another extension related to the CCD would be using the CCDI with axial points outside the design space for problems where feasible. Again due to the lack of success for the CCD design this is a lower priority extension.

Beyond the simple extensions regarding CCDs would be incorporating more designs such as orthogonal Latin hypercube designs and a variety of orthogonal arrays. Also comparing these designs effectiveness to sequential designs would be valuable. Further, we
would like to get a sense of how much value is gained by using a sequential design over an appropriate (optimal) one shot design.

The extension which we think is likely most valuable would be looking at the effect of using higher order regression models in UK. This might vastly change result, most specifically making CCD more likely to be valuable since they have second order models in mind. Another change to our metamodeling technique would be investigating the effects of using other covariance functions.

Our results demonstrate the difficulties for any “cookie cutter” solutions. Each problem is unique and deserves its own metamodeling and design. An analyst must weigh many complex factors in making these decisions. The most prominent factors in the decision are: the use of the metamodel (speed, purpose, accuracy, etc.) and the problem characteristics (shape, noise, and dimension). For each question answered others seem to then spring up in its place. This is why each situation must be considered carefully if we are going to get the fullest value from our limited simulation budget.
CHAPTER 3: REDUCED RANK KRIGING:
DEMONSTRATION AND EXPLANATION

3.1 Abstract

This section will introduce and explain an alternative method for kriging when there is noise in the data, reduced rank kriging (RRK). RRK is based on using a reduced rank decomposition which artificially smoothes the kriging weights similar to the standard spatial statistics approach, a nugget effect. Our primary focus will be showing how the reduced rank decomposition propagates through kriging empirically. Lastly, we will show further evidence for our explanation via other tests. In total, we provide a complete argument for why RRK can is a useful tool for improve SM.

3.2 Motivation

As covered in the introductory chapter, SM requires that we seek maximum predictive value from limited, noisy data. There has been tremendous study of this problem as researchers look for what the “best method” is to create a metamodel. Most of this work focused on comparing one distinct method to another (again we refer the reader to the introductory chapter). In this work we will look at a way to make the commonly used kriging, method better. This improvement is novel, initially counterintuitive, and technical.

To preface the explanation of the methodology, we will initiate the presentation as a thought experiment which will help illustrate a situation where metamodelling can be
improved by the careful analysis on the method itself. In particular this experiment will show two things. First, the thought experiment will show adding an effective “tuning” parameter can improve the performance of a metamodel. Second, it will show how tuning the balance between using local data versus more global data affects predictions.

3.2.1 A Thought Experiment

This example will give the reader some intuition about why and how the using a reduced rank decomposition can be an improvement to standard kriging. A similarly transparent illustration cannot be constructed using kriging itself because the mathematics of kriging are not easy to visualize and understand intuitively.

The thought experiment focuses on metamodeling a one dimensional, quartic function which has been used in literature (Kleijnen & Van Beers, 2004; Van Beers & Kleijnen, 2003). The function is Equation 14 and shown below in Figure 7 as the blue line.

\[ y(x) = -0.0559x^4 + 1.11x^3 - 6.845x^2 + 14.1071x + 2, x \in [0, 10] \quad \text{Equation 14} \]
Further, we assume that we are receiving our data via a simulation and that there is some i.i.d. noise in the data at each point. For this thought experiment the noise is uniformly distributed with mean zero. This noise is added to the function is shown in Equation 15.

$$y_{data}(x) = y(x) + Unif[-a, a], \quad a \in [0, \infty)$$  \hspace{1cm} \text{Equation 15}

Also in Figure 7 are three separate realizations of the noise (when included in the data). These three realizations are shown as the plus signs, asterisks, and circles. The limits of the noise for these three realizations are -3 and +3.

Figure 7: One Dimensional Function and Noisy Realizations
For this thought experiment we will assume that we have data from 26 points evenly spaced along the domain \((x = [0.0, 0.4, 0.8, ..., 9.6, 10.0])\). The particular number of points is important to the results because the magnitudes of the errors in prediction from our metamodel will change. However, the conclusions of this thought experiment (local versus global approximation) is the same regardless of the number of points.

Given the data process as outlined, next consider the task of building an optimal metamodel of a certain type. The type of metamodel to be used in this thought experiment is a moving window regression. For this metamodeling technique, each point to be predicted is found via regression based on the nearest \(w\) points (\(w\) is the size of the window). To keep this presentation as intuitive as possible, we will fit a linear model to the data. In the case of a window of size one, the model will have slope zero and simply use the nearest data point’s value). For all larger window sizes the local metamodel will be of the form \(ax + b\).

This metamodel is obviously a poor representation. However, we will proceed with it because it demonstrates the principles which will explain using RRK later. Picking a moving window size allows the analyst to have different metamodels without changing or adding to the data in any way. Similarly, using a RRK will give us a tuning parameter in kriging, allowing us better predictions.

In the following figures we show the metamodels resulting from various window sizes. Each plot contains the true function (the blue line), the noisy data (the blue circles), and the metamodel’s predictions at 1,000 points over the domain (the red dots which merge into a nearly solid line).
Figure 8 shows the metamodel resulting from using a window size of one, which clearly is a poor fit. First, there is the problem that individual points are subject to too much noise to allow for a good local fit by them even if the function is reasonably flat locally. For example the regions around $x \in [4.6, 5.0]$ and $x \in [7.4, 7.8]$ clearly can be fit well by a flat line. However, the data values received are well away from the true function, so the metamodel does not fit there well. Second, by only using one point, our metamodel does not vary over each moving window sub-domain. While this is not inherently bad, when compared to the metamodels using higher window sizes, it clearly can be improved upon.

Figure 8: Moving Window Regression Metamodel, Window Size = 1
In Figure 9 we see the metamodel resulting from using a window of size two. This choice gives us a metamodel which while not as good as those to come, is a tolerable model. This model has the added benefit that it is piecewise continuous over the entire space, but that feature is not of immediate concern. Currently, we are only interested in the sizes of the errors of predictions from this metamodel. Since there are only two points of data for each local submodel, noise on either point can vastly degrade the quality of metamodel. The errors from this window size are smaller (than window of size one), but still noticeable.

Figure 9: Moving Window Regression Metamodel, Window Size = 2
Using a window of size three is the best model so far (shown in Figure 10). By using three points to estimate the regression lines we begin to protect our model from individual errors in data points. However one outlier point can skew the sub-domain. The average error of prediction over the domain is .8628

Figure 10: Moving Window Regression Metamodel, Window Size = 3
In Figure 11 the metamodel resulting from using a window of size 4 is shown. This representation is our best metamodel so far (and to come). We see an excellent fit near the edges of the domain, and the best fit through the middle so far. By using four points for each local regression model the metamodel continues to become more robust against the noise in the data. This is because (as promised by the law of large numbers) various errors tend to cancel each other out in average. This leads to the parameter values of the metamodel approaching their expectations; the noise gets filtered out. This is also evident in that the average error of prediction over the domain is .7033
In Figure 12 we show the metamodel for using a window of size five. This metamodel has a good fit through various portions of domain \((x \in [2, 3.5] \cup [6, 9])\). However, the fit around the boundaries is beginning to degrade (also near 4). This deterioration of the fit on the boundaries is caused by using data from far away. For example, with a window of five data points, the first sub-domain is over the space \(x \in [0, 1]\) and it uses the points \(x = \{0.0, 0.4, 0.8, 1.2, 1.6\}\) for regression. A similar situation occurs in the final subdomain. Looking at the true data process (noise free), it is intuitively clear that we are using data from parts of the function which exhibit different behavior.

Figure 12: Moving Window Regression Metamodel, Window Size = 5
In the next figure (Figure 13) we see the metamodel which results from using a window of size nine. This metamodel is worse almost everywhere compared to the metamodel using a window of size five. The larger window still protects from individual data point noise, but not significantly more than it already was at window size five. However, using data from too far away is exacerbated. Overall, the metamodel does reflect the proper shape but all the variation is turned down. Since the metamodel uses about one third of the data to create each submodel, the final model is much less varied than the true surface. For example, the first subdomain regression \(x \in [0, 1.8]\) is nearly flat and a very poor fit.

Figure 13: Moving Window Regression Metamodel, Window Size = 9
In the final figure of the series (Figure 14) we view the metamodel when the whole data set fits into one window. We see that this does not yield a good fit aside from some small local subdomains. This progression is the natural conclusion of the problem stemming from using window sizes that are too large. While the large amount data ensures that the presence of noise is less problematic, the large domains mean that we do not have good local approximations.

Figure 14: Moving Window Regression Metamodel, Window Size = 26
Figure 15 shows the residuals of predictions over various window sizes in three ways: first, by average absolute value of residuals, then by the average of the residuals squared, and finally by the maximum absolute residual. In all three criteria the window size of four is optimal (minimal).
We must be careful not to over generalize from a single example. So we repeated this process many times to ensure that these results were typical. In Figure 16 we show the average magnitude of residuals for 1000 realizations of the noise. We also include confidence limits which used Bonferroni’s Inequality and 95% overall confidence. These C.I. limits show that we are comfortable (due to lack of overlap) with the conclusions of this thought experiment (covered next).

Figure 16: Residuals (1 Norm) over Widow Size, Many Realizations of Noise
Figure 17 shows a zoomed in view of the area in the lower left corner of Figure 16.

The preceding series of figures (Figure 8 through Figure 14 and Figure 15 through Figure 17) illustrates a few ideas which we will utilize later. The first is that for a metamodeling method, using a tuning parameter can vastly enhance or damage its predictions. In this case we found that a window size of four to give us metamodel predictions with minimum errors. In most metamodels there exist adjustable parameters which can be tuned by the analyst to achieve best results.
Second, the thought experiment demonstrated that finding the balance between “localness” of the data used and quantity of data used is critical. We showed that with highly local sub-domains, the noise in our data ruins our predictions. Similarly, we showed that more global sub-domains safely filtered the noise from the data but using data from unrelated areas did not add useful information. Later, when we cover how RRK improves kriging, these two ideas will provide intuitive grounding to the explanation.

3.3 Literature Review on Numerical Methods for Kriging

The main focus of literature on kriging’s calculations has been on how to deal with data sets which are too large. This is not a problem for our study since we are working with SM in mind. However, we will cover some of the points in this literature which are nearby our work.

There have been many approaches to the problems of dealing with large spatial data sets. Work on these issues can be traced back to almost the first usage of kriging. This is because matrix inversion has always been an issue which is computationally expensive. As computing power increases, the size of data sets we can analyze grows and there is always another slightly too large set just over the horizon. Hence how to deal with large data sets has always been an active topic. Further, the emergence of “cheap” remote sensing data has created a glut of big spatial data problems for analysts. Before getting into the problems of how to best calculate the prediction from kriging, analysts have wanted to find the best
values for the covariance parameters. This problem has received significant study (Fuentes, 2007; Vecchia, 1988; Whittle, 1954).

Once the proper covariance parameter values have been found, the calculation of kriging weights remains a formidable obstacle. The first set of papers which try to address this issue utilize the idea that much of the covariance matrix will be filled by very small entries. This is because many points in a large data set will have covariances near zero. This means using the sparsity of the covariance matrix can be very valuable (Barry & Kelley Pace, 1997). Also analysts can try to ensure sparsity via covariance tapering (Furrer, Genton, & Nychka, 2006; Kaufman, Schervish, & Nychka, 2008; Shaby & Ruppert, 2012; Zhang & Du, 2008). Another approach to dealing with large data is using a fixed rank approximation is fixed rank kriging, which can handle gigabyte sized data sets (Banerjee, Gelfand, Finley, & Sang, 2008; N. Cressie & Johannesson, 2008). Stein (2008) gives an interesting approach where a sparse global model is then enhanced by lower rank local models to provide great speed and easy parallelization of effort. Zhang and Wang (2010) demonstrate the effectiveness of a hybrid local kriging method which incorporates cross validation. Sakata et. al. (2004) demonstrate another method for reducing the computational burden of kriging. Their method is based on a mixture of formulas and approximations. They balance computation cost with accuracy. They demonstrate a 30%-48% reduction in computational time. Similarly, others have looked for, and found other creative ways to reduce the computational costs for large data sets. (Aune, Simpson, & Eidsvik, 2014; Lindgren, Rue, & Lindström, 2011)
The stream of literature we are more interested in are those papers which have looked at how to modify kriging to improve it for various purposes. This is where our work best fits. Specifically we are interested in papers which looked at modifications to kriging intended to enhance the predictions when there is noise present in the data. The standard geostatistical approach to this problem is to include a nugget effect in the covariance function (N. A. C. Cressie, 1993; Isaaks, 1989a). The nugget effect was developed in spatial statistics to account for independent noise on each data point. In the spatial statistics context this may be measurement error due to inaccurate readings or lack of precision and rounding. It derives its name from mining where even at very nearby sites, response data values may vary substantially; particularly in the context of the presence of valuable ores like gold. (N. Cressie, 1990) In our SM applications it has direction correspondence to the noise present on our simulation data’s estimates. This is a sensible and good initial approach, however there have been improvements on it.

The first paper which examines how to improve over a simple nugget effect is by Sakata et. al. (2007). They propose noise-resistant, smoothed-kriging which is shown to improve on OK with and without a nugget in the presence of noisy data. They do substantial testing on a one-dimensional example to demonstrate the effectiveness of their method. They then follow that up by showing their method’s suitability to a three dimensional optimization problem.

The other paper which is a clear competitor to our method is stochastic kriging by Ankenman et al. (2010). Their work has been inspired by operations research SM.
leverage the fact that they have variance estimates at each of their data points. This allows them to decompose their covariance and by doing so get better estimates. They have demonstrated this improvement over a variety of papers (Ankenman et al., 2010; Chen et al., 2012, 2013; Liu & Staum, 2010; Staum, 2009). The work in this paper will be compared against their method.

3.4 Analytical Modifications for RRK

The place in kriging that RRK impacts is in the inversion of the covariance matrix (which is then used to calculate the Kriging weights):

\[
C_{\text{aug}} \begin{bmatrix} w \\ \mu \end{bmatrix} = c_0 \rightarrow \begin{bmatrix} w \\ \mu \end{bmatrix} = C_{\text{aug}}^{-1} c_0,
\]

where \( C_{\text{aug}} \) is the augmented (by ones and a zero) matrix of covariances from Equation 9, \( c_0 \) is the augmented (by a one) vector of covariances between \( x_0 \) to the data, and \( w \) are the weights which eventually be applied to the data, \( y \).

\[
C_{\text{aug}} = \begin{bmatrix} C_{nxn} & 1_{nx1} \\ 1_{1xn} & 0 \end{bmatrix}, \quad C_{nxn} = [\hat{c}_{ij}], \quad c_0 = [\hat{c}_{i0}]
\]

In Figure 18 we show a modified version of Figure 3 to see visually where this change takes place. Instead of simply taking the inversion for granted, we are now looking at the method itself. While normally this type of analysis would be more in line with numerical
analysis and concerns of stability and accuracy, we will be focused on how adding a well-placed tuning parameter can improve performance.
Figure 18: Modified Kriging Flow, Step One
The difference between Figure 3 and Figure 18 (the arrow from the $C_{aug}$ Matrix now attached to a box for “Decomposition Method”) is important. It is important because while there are many alternative methods for calculating the inverse of a matrix, but as outlined in the literature review, we are focusing on the changes in decomposition method for the $C_{aug}$ matrix and how that then affects the results. It is possible that other choices of inversion methods could also bring benefits; however a tuning parameter is needed before any improvement is likely.

There are few standard options for navigating the “Decomposition Method” box in Figure 18. Possibly the most common is a combination of partitioned inverses and Cholesky decomposition and then direct solving/inversion. Choosing a Cholesky is a natural choice since the non-augmented covariance matrix is positive definite. As a rule of thumb when a matrix is positive definite, a Cholesky factorization is a good choice. Choosing a Cholesky factorization will result in the equations below.

$$C = LL^T, \quad C^{-1} = L^{-1}L^{-1}$$

$$w = \left( \hat{c}_0 + 1_{nx1} \left( \frac{1 - 1_{1xn}C^{-1}\hat{c}_0}{1_{1xn}C^{-1}1_{nx1}} \right) \right)' C^{-1}, \text{where } \hat{c}_0 = \begin{bmatrix} c_{10} \\ \vdots \\ c_{1n} \end{bmatrix}$$

However, we will change that decision to singular value decomposition (SVD) and then perform the inversion from there. Originally, this change was done in hope of gains in computation time and storage in exchange for minor losses in the quality of predictions. However, it was quickly discovered that such gains range from non-existent to negligible.
Fortunately, it was also found that doing this could provide surprisingly good predictions, in fact sometime better than standard approaches.

It is very important to remember the covariance function being used to fill out $C$ is the one found without a nugget effect. As explained in section 3.3 including a nugget effect is reasonable in SM since we expect there to be some amount of noise on each data point from our simulation. Accounting for this noise will produce better metamodels than if we treat the data as deterministic and exclude any nugget effect. However, the reduced rank decomposition creates a similar effect as a nugget term and so that term will not be needed in RRK. The explanation for how it creates this “artificial nugget effect” will be outlined in section 3.6.

Here we proceed with the analytical development of the method. The SVD is shown below, where the singular values in $\Sigma$ are ordered with the largest in the upper right.

\[
C_{aug} = U\Sigma V^T, \quad C_{aug}^{-1} = V^{-1}\Sigma^{-1}U^{-1} = V\Lambda^{-1}U^T,
\]

\[
U = [u_1 \ldots u_{n+1}], \quad V = [v_1 \ldots v_{n+1}]
\]

\[
\Sigma = diag(\sigma_1, \ldots, \sigma_{n+1}) = \begin{bmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_{n+1} \end{bmatrix},
\]

Since we are using a SVD, $U$ and $V$ are both orthogonal. Using a full rank SVD results in the same kriging weights as using standard inversion methods up to standard
numerical precision. Our interests lie in using a reduced SVD. Let $U_r, V_r,$ and $\Sigma_r$ denote the reduced rank versions of $U, V,$ and $\Sigma$ respectively, as shown below.

$$U_r = [u_1 \ldots u_r], \quad \Sigma_r = \begin{bmatrix} \sigma_1 & \ldots & 0 \\ 0 & \ddots & \sigma_r \end{bmatrix}, \quad V_r = [v_1 \ldots v_r],$$

Reduced SVD (RSVD) has been well known in signal processing and statistics. For us it will create an artificial nugget effect which will help smooth the kriging weights. $U_r, \Sigma_r,$ and $V_r^T$ are $(n + 1)rxr, rxr,$ and $rx(n + 1)$ respectively. When multiplied together they will form the proper low rank representation of $C_{aug}$.

$$U_r \Sigma_r V_r^T = [u_1 \ldots u_r] \begin{bmatrix} \sigma_1 & \ldots & 0 \\ 0 & \ddots & \sigma_r \end{bmatrix} [v_1 \ldots v_r],$$

$$C_r = U_r \Sigma_r V_r^T, \quad C_r^{-1} = V_r \Sigma_r^{-1} U_r^T.$$

Now with our slightly (or possibly significantly) different $C_r^{-1}$ we get different kriging weights. Figure 19 illustrates where our focus now shifts; the box “Kriging Weights.” These kriging weights are simply the product of using this different C inverse with the same $c_0$ vector as before.

$$w_{original} = L^{-1} L^{-1} c_0, \quad w_{RRK, r} = V_r \Sigma_r^{-1} U_r^T c_0.$$
3.5 Methods

The testing and conclusion in this chapter are limited to the experiments provided. It is impossible to test every possible situation, but we have endeavored to test these results in a wide representative variety of situations. We have found such consistency in the results that we have great confidence in the robustness of our conclusions. For the bulk of this chapter we will focus on a single example from beginning to end. After we cover this one example in depth we will give some other examples which show the results of RRK without going into each specific process.

To pick the placements for our points we refer back to our results from Chapter 2 and used a Latin Hypercube Design. This design is robust and generally a very good choice in many applications including those of higher dimension. We then generated data from two different functions. Remembering our design space is the unit hypercube, we define the functions as follows:

- A second order polynomial without any interaction terms. It has random coefficients in each dimension

\[
y = ((2X - \mathbf{1}_{nx1}) \cdot (2X - \mathbf{1}_{nx1})) \mathbf{\beta}(\cdot,1) + (2X - \mathbf{1}_{nx1}) \mathbf{\beta}(\cdot,2),
\]

\[
where \mathbf{\beta} \text{ is a } d \text{ by } 2 \text{ vector and } \beta_{ij} \sim Uniform[-1,1]
\]
• A rounded wall, a function with explosive growth on one boundary for each
dimension ($\frac{1}{1.1-x}$ signifies element by element division).

$$y = \left(\frac{1}{1.1-X}\right)\beta(:,1) + (2X - 1)\beta(:,2),$$

where $\beta$ is a $d$ by 2 vector and $\beta_{ij} \sim Uniform[-1,1]$

We are focused on situations with limited but not severely limited data (100-400
points) in relatively low dimension (2-5). We wanted problems which were not particularly
difficult and were computationally fast. For our detailed example we will take an instance of
problem two with two dimensions (two enable visualizations) and 250 data points.

Next we added random noise to the simulated data. To do this we followed a common
process of creating replications (Ankenman et al., 2010; J. P. C. Kleijnen & W. C. M. van
Beers, 2005). For each micro replicate we took the true data value $y(x_i)$ and multiplied it by
a random normal number with mean 1 and standard deviation of 0.2. Finally, we averaged 25
of the micro replicates to get our simulated data value for that point (plus a variance estimate
which is needed for SK).

We then used universal kriging on simulated data via the software package DACE
(Nielsen, 2002). For modeling, we used regression models of order one. We used an
exponential covariance function in the product form to allow for anisotropy (Isaaks, 1989a).

The set of tests outlined are not comprehensive, but they have given us valid,
consistent results and are a reasonable place to start. Issues regarding the generality and
applications of these results are topics we expand on in the conclusion of this chapter and further in Chapter Four. However, overall, the results presented here are consistent and encouraging.

3.6 How RRK Propagates through Kriging

We measure the success of a metamodel by the size of the difference between the predicted value and the true data value of the point to be predicted at (the residual). To understand how utilizing an RRK can enhance kriging we will track how this change propagates to the residual. First, we must look at exactly how the reduced rank decomposition enters into the kriging process.

One’s first instinct might be to simply look at various $C_{aug}^{-1}$ matrices and try to ascertain what happens. However, since there are $(n + 1)^2$ entries in each $C_{aug}^{-1}$ matrix, discerning a pattern can be difficult and potentially misleading. As mentioned in the literature review, we are interested primarily in investigating using an RSVD’s effects. The place in kriging that RRK first impacts is in the inversion of the covariance matrix (which is then used to calculate the kriging weights).

$$C_{aug} \begin{bmatrix} \mathbf{w} \\ \mu \end{bmatrix} = c_0,$$

$$\begin{bmatrix} \mathbf{w} \\ \mu \end{bmatrix} = C^{-1}c_0$$
Figure 19: Modified Kriging Flow, Step Two
The change in the weights is where reduced rank kriging makes a noticeable and comprehensible difference in the kriging process. Our first question was whether the change would be slackening the unbiasedness of kriging. Kriging does not claim to be the most efficient prediction, only the best unbiased one. So, we checked the sums of the weights. They still sum to one within reasonable numerical tolerance. Further testing the residuals of predictions empirically have shown no to negligible bias.

Next we thought to look at the distribution of the weight. This is where the changes become visible as the rank of the RRK changes. Using a full rank RRK is the same as the standard methods and that distribution of the weights is shown in Figure 20.
In Figure 21 we plot the same histogram, but this time on a log scale so that the variation at the tails (large magnitude weights) can be seen more clearly. We added 1 to all the raw frequencies, so that when the log transform was applied none of the bins transformed to negative infinity. Hence any data point which shows a transformed y value of 0 (canot be seen on the plot since it overlaps the horizontal axis) actually has true frequency of 0, not 1.

Figure 21: Standard Kriging Weights on a Log Scale

Figure 21 only shows the situation for a single problem with a single random Latin hypercube data placement. In Figure 22 we show the average histogram for many replications of problems and data placements. We choose not to show the confidence limits here because they are tight and provide no important additional information.
We observe that as there are both positive and negative weights, but clearly there are more positive ones (as expected). Further, a tremendous number of points get weights very close to zero. While this plot may obscure this second fact, we can refer back to Figure 20 for evidence of this behavior. Combining these two observations yields our fundamental insight about standard kriging: A few points have a large influence on determining a predicted value and most points (orders of magnitude more) only a small influence (or no influence).

Figure 22: Full Rank Kriging Weights, Many Replications on a Log Scale
Next we reduce the rank of the RRK and observe the results. Figure 23 shows what happens when moving down in rank to 176 (from 251). We see that the distribution of weights is tighter. The number of extreme observations (absolute magnitude above 0.4) is less. Also there seems to be a shoulder developing around +0.3.

Figure 23: High (Not Full) Rank Kriging Weights
In the next figure, Figure 24, the distribution of weights is even tighter. This figure shows the weights when using a medium rank RRK; in particular, this one has rank 101 (compared with 251 for the full rank). The number of extreme observations (absolute magnitude above 0.4) is dropping to where they are now very infrequent.

Figure 24: Medium Rank Kriging Weights
In Figure 25 we see the final step in this evolution, with a rank 26 RRK. The weights have become extremely clustered around zero. In particular there appears to be no weights below -0.15 or above 0.25.

Figure 25: Low Rank Kriging Weights
In Figure 26 we show all four histograms into a single plot. The Full, High, Medium, and Low Rank weight distributions are shown in Blue, Blue-Purple, Red-Purple, and Red respectively.

Figure 26: Kriging Weight Histogram Comparison

Figure 26 gives the pattern better focus. As the rank goes from full to low the distribution of weights changes from blue to red. We see clearly that with high full rank representations there are a few large magnitude weights (positive and negative) and many tiny (near zero) ones. As the rank reduces, the large magnitude weight go away and many of the near zero weights become small positive ones. We refer to this as shift in weight distribution the “democratization of weight.” Others might call it a smoothing of weights.
What this means for predictions is that instead of the observations from a few points having virtually all the power in prediction, many points contribute information. It is intuitive to see how in the presence of noisy data, this distribution of weights could provide for better predictions.

This analysis begs the question: which particular data points are getting more (or less) weight as the rank of the RSVD changes? This question is important because simply knowing that the distribution of weight is different does not lead to any useful conclusion. We must know how it is different. This new question signals a transition from looking at the weights themselves to where those weights go. This next consideration is shown in Figure 27.
Figure 27: Modified Kriging Flow, Step Three
To answer this question (where does the weight go?) we took at each point which is predicted individually and then looked at the pattern of the weights which were distributed around it. In particular, we treated each point as if it was at the origin and then positioned the weights which were given to each data point around it. Then all the plots for each predicted point were combined into a single plot which covered the area [-1, 1]x[-1, 1]. There were a tremendous number of points (number of data points*number of test points), so we averaged the results over small subgrids and then constructed a contour plot.

Figure 28 is the result of this process when using the full rank (standard kriging). Before we examine the details of this particular plot, we highlight a few features of the plot. This plot only covers the area [-0.5, 0.5]x[-0.5, 0.5]. This reduction in region occurs for two reasons. First, this shows the area where we have large amounts of data and so the results are somewhat free of noise. When we look at points out on the edges of the [-1, 1]x[-1, 1] space, there are not enough points to obtain clear results. Second the center ([-0.5, 0.5]x[-0.5, 0.5]) is of greatest interest. The important patterns are in that center area, while the outer areas are virtually flat with all weights at (or very close to) zero or they can easily be extrapolated based on the behavior in the center. So by zooming in on the center area, we can easily visualize the important variation of kriging weight placement.
In Figure 28 we see very tight clustering of positive kriging weights very close to the origin (near the points to be predicted at). Also there is a tight band of negative weights just outside the positive bands (dark blue streaks near the light blue, around $\pm 0.02$). The overall pattern (a cross) is a result of the covariance function choice. We choose an anisotropic exponential covariance function without a nugget effect (as explained in section 3.3) which is linked via product form (shown below), and so the shape is expected.

$$C(x, x') = \prod_{i=1}^{d} e^{-\theta_i h_i}, \text{ where } h_i = |x_i - x'_i|$$
If something isotropic was chosen instead, concentric circles would be seen. The seeming variations on the corners of the plot are simply noise and can be disregarded. The points are all in such a tight band near zero that the variation is practically insignificant. Lastly, note the upper limits of the color bar go up to nearly 0.5.

![Figure 29: Kriging Weight Distribution, High Rank](image)

In Figure 29 we see the distribution of weight for a high rank RRK (rank=176). We continue to see tight clustering of positive kriging weights very close to the origin. Also the tight bands of negative weights just outside the positive bands remain. Clearly the plots
(Figure 28 and Figure 29) are very similar. There are two small differences. First, the upper limit on the color bar has dropped slightly. Second, the positive weight areas have expanded a tiny bit.

Figure 30 corresponds to a medium rank (rank=101) RRK. As expected, we again see a drop in the top end of the color bar. Also there is noticeable spreading of the positive color areas, which corresponds to more points having positive nonzero influence. Also we now can see that the points getting increased positive weight are the ones which are close but not very
close based on the covariance function. This behavior is intuitively sensible based on the principles of kriging: specifically those points which are nearby tend to vary similarly. Also we can see that the points losing weight are those which were very close to the origin. This behavior is the democratization of kriging weight. The advantage of the democratization of kriging weight is the same as what was seen in the thought experiment in section 3.2.1. Since there is noise on the data, giving too much voice to a few points allows for local noise to vastly impair the metamodel. However giving other (slightly) less nearby points greater weight reduces the negative effects of noise.

Figure 31: Kriging Weight Distribution, Low Rank
The final figure in the series corresponds to a low rank (rank=26) RRK. For this rank the top end of the color bar has dropped precipitously. Also the spreading of the positive weight area continues; the positive weight area covers a much larger space than in any previous figure. We also see the negative weight area (dark blue areas) has spread out more relative to the small tight strips it originally covered. This change is the natural progression for the democratization of weight. Many points are having small influence in the prediction and no points, not even those very close to the origin, have large weights. This is analogous to the ending Figure 14. There comes a point where we are likely using data which is far too far away, instead using less (and more local) data would be better. In essence, RRK’s metamodel is overemphasizing global information and underemphasizing local information.

We now have an answer to the question of which points have greater and lesser influence. Points very close to the point to be predicted actually receive less positive weight than they started with as more points get a positive weight. On the other hand, points which are near to the origin, but not very close, go from having nearly zero weight, to a small negative weight, to finally a small positive one.

Reassuringly, we now can see that the points getting increased positive weight are the ones which are close based on the covariance function. This observation is intuitively reasonable based on the principles of kriging: points which are nearby tend to vary similarly. This behavior explains how RRK improves the predictions on problems with noise as it helps reduce the impact of noise on the predictions. Now we have a concrete, though experimental, understanding of the redistribution of weight relative to the point being predicted.
3.7 Results

In this section we will focus on the predicted values and their associated residuals from RRK while comparing those to the predictions from standard kriging. Visually this change of attention is illustrated in Figure 32.
Figure 32: Modified Kriging Flow, Step Four
3.7.1 RRK Residuals as a Function of Rank

In Figure 33 we show the residuals for an example problem as a function of the rank of the RRK used. The rank increases from left to right along the horizontal axis, with full rank on the right end and very low rank on the left. The horizontal axis (which is covered by the residuals) corresponds to a residual being zero (perfect prediction) for that rank. In this plot the best rank to use is the one which best "bunches" the residuals near the horizontal axis.

![Figure 33: Residuals as a Function of RRK Rank](image)
It is interesting to see that the evolution of each residual is quite noisy. Each residual trajectory often bounces up and down many times along the way. However, the overall trend towards the horizontal axis and then away from it is clear. This behavior indicates that while for some rank RRK may improve predictions in general, for a particular point to be predicted the optimal rank may be almost anywhere. Further, we see there may be many local optimal ranks for a given predicted point. These issues will complicate the concerns in Chapter Four.

Figure 34 shows the same basic information as Figure 33 except we plot the absolute value of the residuals. This helps make it clear that there is some rank, well below full rank, which is better than full rank. This plot also makes it clear that there is some point at which the rank becomes too small, and the predictions become much worse.
The first diagnostic we need to check for this new method of kriging is unbiasedness. There are two ways to check this. First we can see if the weights still sum to one. After substantial testing, they still do. Second we can look at the residuals of predictions and their sum. If the predictor is unbiased we expect that the mean of the residuals would be zero. In our testing we find that the confidence interval of the average value of the residuals is contains zero. Hence we think this modification (RRK) is unbiased. Now we check for the efficiency of RRK’s predictions.

We have made the argument that using RRK can improve residuals (in particular we said more can be near 0). However, we need greater understanding of how this clustering occurs. To obtain a greater understanding, we will plot the distribution of the absolute value of residuals. Figure 35 shows the distribution of the absolute value of residuals for full (251), high (176), medium (101), and low rank (26) kriging in black, blue, green, and red respectively. The plot also contains confidence limits (each at an individual 95%) to show that sufficient replications have been done such that there is confidence in all results.

It is important to note that this plot is on a log scale vertically. By using a log scale we can easily see that at full rank the tail of the distribution is exaggerated. It is easy to determine why low rank is not as good a choice as the others. Low rank has many large residuals because it has given up too much information; its rank is too low. The interesting comparison is between the cluster of medium, high, and full rank.
In Figure 36 we show the “relative CDFs” of the medium, high, and full rank metamodels. For this we take the distributions from Figure 35 (which are PDFs) and then plot their CDFs. However the distributions are so similar, that differences between the CDFs are tough to ascertain. So we then subtract the average value over all three CDFs at each point along the true CDF and then plotting the three resultant curves. A particular curve being higher than the others implies it is probably better at least up until that threshold residual level. Being higher vertically implies that it has a higher proportion of its residuals below the given threshold.
We see that all three CDF come together beyond 1.5. Hence they all behave the same in terms of having large residuals. Hence the effect of the RRK shows up on residuals below 1.5. As explained earlier, the curves which go the highest perform the best. This gives great evidence that medium rank RRK, is better than high rank RRK and in turn full rank kriging. However, since a few residuals to the right of Figure 35 can vastly outweigh many points on the left we created Figure 37.
Figure 37 shows the same histogram as in Figure 35 with a couple of major changes. First, each point from Figure 35 is now scaled by its importance, specifically it is scaled by its location along the x axis. The second change is that things are now plotted on a linear scale. These two changes allow for a useful property: easy comparison. The area under each curve is equal to the 1 norm for each rank level. So now we can compare the quality of each rank level by eye.

From Figure 37 it is again clear that low rank RRK is vastly inferior to the other three options. Also we can now see clearly why medium rank beats out high and full rank. It moves a small proportion of its residuals of sizes 25 to 1.5 down to below 0.25. So now we
have our result for what RRK improves: it moves a small fraction of the medium and large residuals down to being small ones. Based on our knowledge from Figure 33 and Figure 34 we know that in reality probably what is happening is that the residual at one point goes from 1.2 to 1.0, another from 1.0 to 0.9, another from 0.9 to 0.5, and another from 0.5 to 0.2.

3.7.2 Various Norms for the Residuals of RRK Predictions

There are many ways of expressing the quality of predictions. So far we have given as comprehensive a study of the residuals as appears in literature. The most common method of summarizing the residuals in literature is quantifying them by a single number, particularly norms. We showed some examples of this in our motivation section earlier. There are three norms commonly used: one, two, and infinity. Each one can show a different facet of the quality of prediction. In the following figures we show the one, two and infinity norms for the example we have been working with respectively.

For all three figures (and those to come later) the horizontal lines are the respective norms for the standard methods of metamodeling: universal kriging without a nugget effect (in green), universal kriging with a nugget effect (in blue) and stochastic kriging (in red). These lines are horizontal because these techniques have no tuning parameter based on rank, and so do not vary over the horizontal. There are confidence limits shown for each of these horizontal lines in each instance as well. We are not currently interested in the interplay
between these lines with each other; they are serving as standard technique baseline against which we can demonstrate RRK (in magenta).

These plots were made by running 500 independent replications of data placement and problem parameters. Then we compiled the results from those and were able make confidence limits. Again these confidence limits each have individual confidence of 95%. Finally we note that RRK converges on the right to the green horizontal line in every plot. This is confirmation again that when full rank is used in RRK, the results are standard Kriging. The reason for the convergence on UK without a nugget effect and not another is that we used the covariance from that model for the RRK.
In Figure 38 we see the 1 norm values. These are useful because they give a sense of the average error (MAE) on any point picked at random. If we divide the 1 norms by the number of test points we get precisely the MAE. We see that both RRK methods outperform the UK from which they draw their covariance function from over almost the entire domain of potential ranks. Further, the RRK beat out UK with a nugget effect over a limited but noteworthy domain ($r \in \sim[35, 125]$). However neither of the RRK methods outperforms SK anywhere over the domain. They do come tantalizingly close near their optimum $r$ (around 50), but they come up just short.

Figure 38: 1 Norms for Various Ranks of RRK and Baseline Methods
Next we turn our attention to the infinity norm (Figure 39). This measure sheds light on the Maximum Absolute Error (ME). It may be surprising that we study the infinity norm before the 2 norm, but doing so will help make the 2 norm more clear. Again we see the RRK perform better than UK without a nugget effect. This time the dominance is not over as large a part of the rank domain, but it is still the vast majority of the domain. Interestingly, now the UK without a nugget effect outperforms both UK with a nugget and SK. Hence when RRK dominates UK without nugget, it in turn dominates the other two baseline choices.
Figure 40 shows the 2 norm for each method. This figure is in many ways an average figure blank and figure blank. Again the UK without nugget model is soundly defeated by RRK. Also the UK w/nugget and SK are beat over a reasonable part of the domain. These experiments show that there clearly are times where using RRK improves predictions.

We want to expand our experimental results. First we will look how RRK performs on a different problem (problem 1 with 3 dimensions). We will repeat the testing parameters (500 independent replications, same amount of noise and data points) and see the results.
Figure 41 shows the 1 norm. Again RRK outperforms the UK without nugget. SK is better than the other two baseline methods. Again RRK covers much of the gap from UK to SK but it comes up a bit short.
Figure 42: 2 Norm for Various Kriging Methods, Quadratic Problem

Figure 42 shows the 2 norm. As we have become accustomed RRK outperforms the UK without nugget. Unlike in the other problem, RRK does not outperform SK.
Figure 43 shows the Infinity norm. This looks very much like the 2 norm figure (Figure 42). RRK outperforms UK without nugget and covers much of the gap to SK, but it does not outperform SK. These figures give further confidence that RRK improves on the performance of UK without nugget. It does this in a way where it consistently best UK w/ nugget (the geostatistical method for this process) and sometimes outperforms SK (the OR method for this process). Next we will look at two factors which impair RRK’s performance relative to UK without a nugget.
3.7.3 Effects of Dimensionality, Noise, and Number of Data Points on RRK

In Figure 44 we show the average norms (1, 2, and infinity) of the residuals of prediction for 500 instances of 7 dimensional version of problem 2 with 100 data points. We only show the results from UK without nugget and RRK for the sake of clarity. Also for this figure as well as the three that follow it we no longer show the norms in absolute terms, we instead scale them all to be 0 to 1. In Figure 44 we see that the RRK does not outperform the UK on any of the norms, which demonstrates that there are cases where RRK *does not* improve the predictions from UK without nugget.

Figure 44: Norms for 7-Dim Problem with 100 Data Points
In Figure 45 we show the average norms for 500 instances of 7 dimensional version of problem 2 with 400 data points. Again we show the results from UK without nugget and RRK. In this case it is clear that RRK does outperform UK on all three norms.

Figure 45: Norms for 7-Dim Problem with 400 Data Points
In Figure 46 we show the average norms for 500 instances of 2 \textit{dimensional} version of problem 2 with 100 data points. Here also RRK outperforms UK.

![Norms for 2-Dim Problem with 100 Data Points](image)

**Figure 46: Norms for 2-Dim Problem with 100 Data Points**

From the previous three figures, our findings support that when problems are particularly “difficult” RRK does not help. We use the term difficult to imply a lack of data for the complexity of the problem. Specifically, this means that the ratio of dimensionality of the problem is too great for the quantity of data points. In the second and third figures we demonstrated that as the number of points grew or the dimensionality of the problem dropped...
(the problem became “easier”) RRK was able to again surpass UK. This is not to say that these problems are truly easy, only that they are easier or harder relative to themselves and other SM problems. However, we do know that there is a relationship between the number of points, the dimensionality of the problem, and the relative improvement found via RRK.

Lastly, we will show what happens in the absence of noise in the data. Figure 47 shows the norm for all the methods (both UKs, SK, and RRK). We did not include confidence limits in this figure for the sake of clarity.

Figure 47: Norms for Different Methods without Noise Free Data
There are two takeaways from this figure. First, without any noise in the data, the UK without a nugget and SK perform the best, and practically equivalently, on all three norms. This is what we would expect since there is no noise UK without nugget properly fits the data in that regard (it still does not have the proper regression model, but neither do the other methods). SK actually outperforms UK without nugget because its extra flexibility can easily account for the lack of noise and even slightly improve over the UK, since both methods use a regression model which underfits the true data. Second, we see that RRK does not perform better than UK without nugget. What this means is that without the noise there is no value in using more points to make predictions. In essence, we should simply rely on only a few nearby points, and that is sufficient. This is the final support needed to show that the reason RRK improves predictions is the democratization of riging weight to protect against noise in the data.

3.8 Conclusions and Extensions

This chapter has presented a novel method for executing a step of the kriging calculation. Specifically, we have shown that using a reduced rank decomposition to go from \( C_{aug} \) to \( C_{aug}^{-1} \) can improve the predictions. The reduced rank decomposition we have employed is reduced singular value decomposition.

The other major contribution of this chapter was an explanation of why RRK works. The force at work is that by using these reduced rank decompositions reduces the magnitudes
of the largest weights (both positive and negative). To maintain unbiasedness many data
points get small positive weight. Instead a few points having all the influence on a prediction,
many points have a small influence. We called this change the democratization of kriging
weight.

There are many exciting avenues for building on these results. First is learning more
about the difference between those problems where RRK will improve the predictions over
UK without nugget and those where it will not. Beyond that is finding where, given there is
improvement, does that improvement make RRK superior to other common methods, namely
UK with a nugget effect and SK.

Closely related to the problem of understanding when RRK is worthwhile is finding
the best rank to use. In practice, an analyst does not know the optimal rank to use. This
problem occurs because they do not know the residuals of the points they will predict. This
means it is important to find a method to search for the optimal rank to use and how to
confirm it has been found. Closely related is being able to find a good estimate of optimal
rank. It is important to find answers to these questions such that this method can be used in
practice. This problem will be a major focus of chapter four.

Another useful result would be to know if certain points to be predicted may have a
particular rank for this prediction. This is in essence finding the optimal rank for each point
to be predicted at individually. Additionally it is worth exploring if these concepts can be
similarly incorporated into other kriging frameworks, most specifically stochastic kriging.
reduced rank kriging shows great promise for use in metamodeling these practical simulation problems. It gives us another way to handle the noise of a simulation. It does this by adding in a tuning parameter which allows for better prediction by democratizing the kriging weights.
CHAPTER 4: OMNI-RANK KRIGING

4.1 Abstract

Reduced rank kriging (RRK) was introduced in the previous chapter and shown to have promise as a simulation metamodeling method. However that chapter left open the question of how to find the best rank to use and the question of under what circumstances RRK is as an equivalent or superior method. In this chapter we address those issues. First, we explore how to find the best rank for RRK and create a modification we call omni-rank kriging (ORK). We then explore how ORK performs against a peer method stochastic kriging, over a variety of circumstances. In the end we conclude the methods are competitive with each other, and which is better is dependent on the specifics of the problem being metamodeled and the measure of interest.

4.2 Motivation and Introduction

In the previous chapter we showed how reduced rank kriging (RRK) can be effective metamodeling tool because its predicted values have residuals of similar and sometimes better size than SK and UK (depending on the norm used). However, we did not explore the question of how to pick the rank for use in RRK such that the promising results can be obtained on an individual problem. This choice is important because we saw that poorly picking the rank used can erase any benefits RRK may have (see Figure 38 through Figure 43).
Finding the optimal (or even an acceptable) rank is a complex problem. We have shown that over many problems using somewhere between 25% and 50% of the full rank achieves good results. However, we also showed that the best rank varies based on the problem and in fact sometimes RRK does not help. In this section we will address these issues.

4.3 Literature Review and Previous Work

There is little previous literature directly applicable to this problem since RRK was developed in the previous chapter and does not have a literature base. The literature we do find applicable is that which examines how to test simulation metamodeling methodologies and on stochastic kriging as metamodeling method.

Much of this chapter will be devoted to comparing metamodels and the techniques which create them. There is a wealth of literature comparing various techniques for us to draw upon (Clarke, Griebsch, & Simpson, 2005; Ruichen Jin et al., 2001; Ruichen Jin, Du, & Chen, 2003; Y. Li, S. H. Ng, M. Xie, & T. Goh, 2010; Stander et al., 2004). Our work will be patterned after those studies by looking at multiple perspectives on the quality of the metamodels created.

We choose to use stochastic kriging (SK) as our ultimate measuring stick for the effectiveness of RRK. SK is a method which quantifies uncertainty into two categories, intrinsic and extrinsic, and by doing so modifies the basic Kriging methodology to effectively
metamodel stochastic simulation data. (Ankenman et al., 2010). SK has been shown to effectively model data (Baysal et al., 2008; Chen et al., 2013; Staum, 2009) and it is an excellent measuring stick since its computational load is similar to RRK and its results are a good target for a similarly complex method.

Compared to many of the metamodel comparison papers we will not have as vast a set of methods or test problems. However, we do introduce a novel metamodeling method. Because of these factors this work is very similar to the papers which introduced SK in structure and content.

4.4 Cross Validation

Our initial plan to solve the rank problem for RRK was to utilize cross validation (CV) to estimate the size of the residuals resulting from different ranks. From the previous section we knew that if we could discover the optimal rank for a class of problems we would have predictions which while not better on every problem instance, could be better on average for some norms. The rank which is best for a class of problems we refer to as the optimal fixed rank (OFR) and its average performance over the class of problems for which it is optimal as optimal fixed rank performance (OFRP). As a quick note, we use the word performance as shorthand for the size of the residuals of predictions for a metamodel. Performance is judged by the norm of the residuals (one, two, or infinity).
Finding a CV procedure which could outperform OFRP was a minimum goal of performance. This goal is important is because on each individual problem, the true optimal rank varies. Often the optimal rank for a given problem will be near (or exactly) the OFR, but sometimes it will not. We hoped that CV could take advantage of this variation and achieve results at least as good as the OFRP.

The second level performance goal was to reduce the optimality gap to the individual problem optimal rank. The individual problem optimal rank performance (IPORP) is the performance of RRK if on each problem and data placement, the true optimal rank for the specific problem is used (as opposed to any fixed rank). The gap between any CV procedure and IPORP is our optimality gap. Hence, all our CV procedure testing focused on reducing the gap to IPORP and checking if our CV performance was superior to that of OFRP.

To test the efficacy of CV for finding the RRK rank and establishing the performance of RRK overall, we created many instances of a few test problems. For each instance of a test problem we created data by creating a RLH and then fit metamodels. We then calculated the norms of the residuals of the predictions for each metamodel.

We tried many CV procedures and found the effort ultimately to be fruitless. This difficulty was caused by the twin problems which make simulation metamodeling so challenging in the first place: lack of data and noise in the data.
In Figure 48 we show for an example problem the results from CV and the true results from RRK. At first inspection it seems that CV does a fine job of approximating RRK’s true performance. The fact that it is consistently off by a noticeable amount is not a problem, so long as the difference is (approximately) consistent over the whole rank space. It seems that the difference is rather consistent on the 2 norm, and varies a bit more for the 1 norm. However, there is another figure which provides vastly greater insight.
In Figure 49 we plot the true RRK performance as a function of CV’s estimate for the example problem. These values are plotted for each rank in the rank space. In a perfect world these points would all be collinear and preferably have a slope near 1. Again at first glance we might think the result is promising, as the rho value is high (~.976 and ~.996), but this is misleading. The apparently high rho is skewed by a few influential points in the top right of the figure. Furthermore the information in the top right is nearly irrelevant to our problem. Points in the top right correspond to CV predicting that certain ranks would be awful for use. While identifying these ranks as terrible is better than identifying them as promising, it is not
particularly insightful as avoiding those ranks (very low ones) is easy. The important part of the figure is in the lower left; down there (shown zoomed-in in Figure 50) are where the true optimal ranks are and how CV estimates their performance.

![Figure 50: Zoom-in of Previous Figure](image-url)

In Figure 50 we see that the relationship between CV estimates and true RRK performance is not nearly ideal. In fact, it is so poor as to make naive CV not viable as a method for picking the rank for RRK. This result occurs because through testing we found
that naïve CV is outperformed by OFR. In fact, naïve CV is outperformed by many potential ranks such that there is little difficulty in picking a fixed rank which outperforms CV.

![Figure 51: Fixed Rank RRK versus CV Performance](image)

In Figure 51 we show the performance of RRK for various fixed ranks (plotted as dots) and the performance of naïve CV (plotted as the dashed line). We also zoom-in on the overlapping areas in Figure 52. This behavior is a representative example in that picking any rank between 30% and 45% of full rank will be better than naïve CV.
In Figure 53 we expand on and show the results differently. The rows are the one, two, and infinity norms respectively. In the left column, we plot the performance of the OFR on many distinct problems (from the same class) against the performance of CV. In these figures (the left column) each problem instance is represented as a point. The x coordinates are the optimality gaps between the performance of CV and the IPORP on each problem. The y coordinates are the optimality gaps between the performance of OFRP (we used rank of 50 since that is good for all three norms) and the IPORP on each problem. We also plot the line

Figure 52: Zoom-in of Previous Figure
x=y. In the right column are histograms of the values x-y from the respective left column subfigures. These sub-figures show the frequency of various differences in performance.

Figure 53: CV’s Performance Compared to OFR’s Performance

These six subfigures are a more complete demonstration of the fact that CV is outperformed by OFR. In the left subfigures we can see there are far more points on the right/below the line than to the left/above it, particularly in the one and two norms. This demonstrates that there many problems where OFR performs better than CV. The right
subfigures further emphasize this point. The positive skew demonstrated in the figures represent that there is much more mass where OFR is better than CV.

We next considered how to improve naïve CV. We considered modifying CV via methods based on Bayesian Information and mixing naïve CV procedures. However none of these avenues ultimately proved successful.

4.5 Omni-Rank Kriging

After concluding that CV was not a promising method for employing RRK, we sought a new, suitable method. The method we created was inspired by two observations. First was the idea that democratizing the kriging weight is what caused RRK to work in the first place. This observation reminds us of the central limit theorem (CLT). If instead of a search for the best singular rank, suppose we look to combine multiple ranks to form a better estimate.

This idea (using multiple ranks and mixing the weights from each) was also influenced by Figure 33. Here the influence of the CLT is even more visible. In that figure (reproduced here as Figure 54) we see many residuals of prediction as a function of rank. In chapter three we focused on how the overall clustering of those residuals varied as function of rank which was a surrogate for the efficiency of RRK as an estimator. Here we consider the evolution of each individual residual "path."
We see that these paths look very much like random walks. Often they seem to exhibit drift and the fact that the small lag autocorrelation is high is inescapable. One fact we know from Chapter 3 is that RRK predictions are, on average unbiased, regardless of what rank RRK is used. This observation is critical for the next step.

If we averaged the residuals over a set of the ranks we would get an estimate which is the average of many unbiased and somewhat correlated random variables. We hoped that this averaging could allow some of the residuals to cancel and improve the predictions.

Figure 54: RRK Prediction Trajectories (Figure 33 Reproduction)
So for those reasons we designed a method which would average the weights from each rank. We call this approach omni-rank kriging (ORK). This approach has two advantages compared to CV methods. First, and most importantly, we will show that it produces better predictions than CV with RRK. In fact we will show that in many cases it even outperforms IPORP. Second, it also saves the computational effort of fitting for CV.

![Graphs showing performance of ORK vs CV](image)

**Figure 55: Performance of CV Compared to ORK’s (1, 2, and Infinity Norms)**

In Figure 55 we compare CV’s performance to ORK’s. We do so using the same procedure as when we were comparing CV to OFRP in Figure 53 (the 1, 2, and infinity norms...
norms are shown as the first, second, and third rows respectively). From this we see that ORK is superior to CV. It is important to note that a number of points have a negative y coordinate, which shows that on those problems ORK outperforms IPOR. This performance was something we did not consider as an achievable goal with CV (and it was not with CV). As a result, ORK not only achieves our goal for acceptability (outperforming OFRP) but it achieves a level of performance which sometimes surpasses what we considered possible.

![Figure 56: Performance of ORK Compared to OFR’s Performance (1, 2, and Infinity Norms)](image-url)
In Figure 56 we again repeat the comparison figures, this time with ORK’s performance as the x coordinate and OFRP as the y. Clearly on the 1 norm ORK is superior to kriging even using the optimal fixed rank. On the 2 norm it seems to be a tossup as to which is better (and it is). However, on the infinity norm kriging using the optimal fixed rank is better. So while ORK does not fully dominate kriging using ORF, it is vastly improved over CV and is a method which is reasonable to compare to a current SM method from literature.

4.6 Omni-Rank Kriging versus Stochastic Kriging

Now the crucial issue is to compare ORK to an independent method, stochastic kriging (as opposed to a version of RRK). SK is an excellent measuring stick for two reasons. First it is a current method which has shown great promise and effectiveness in OR. Second it is a method which requires a similar amount of computation to ORK. As we will elaborate on in detail later, ORK is a computationally expensive method. This computational cost means that for the most part it would be inequitable to compare ORK to simpler, cheaper methods. SK on the other hand, while still cheaper than ORK, it is at the same level of complexity and computation.
To begin our comparisons we start with the same set of subfigures as from the previous section (Figure 57). However, this time we are not showing their performance gap to IPORP because IPOR is not a useful baseline since both methods can outperform IPORP (this explains the major change of scale). In this case the x coordinates are ORK and the y coordinates are SK. It is clear that for this particular set of problems SK is superior on the 1 norm. This result is not particularly surprising since SK has outperformed RRK in the previous chapter. We showed in the previous section that ORK improves over RRK on the 1

Figure 57: ORK’s Performance Compared to SK’s Performance
norm, but it appears ORK does not improve enough to pass SK. However the comparison is much more competitive than RRK (including when using OFR). On the other hand ORK is able to maintain the superiority that RRK (specifically when using OFR) demonstrated on both the 2 norm and infinity norm.

4.6.1 The Base Problem

The results in Figure 57 were restricted to very specific set of problem instances. Through experimentation we have seen factors which effect which method is superior. We now will explore those factors. To compare SK to ORK more extensively we varied the number of points in designs for each problem instance. We then refitted the metamodel with new data points. Again, we find the norms of the residuals of predicting independent points. We then plot the norms as a function of the number of points for each metamodeling method.
In Figure 58 we give an example of this process. For a given problem instance of our “base problem” (QN type problem, 4 dimensions, average noise) we created designs of 16 to 96 points and plotted the norms of the residuals. By comparing the curves vertically we see the differences in sizes of norms for a given number of points. The ratios of the norms of ORK to the norms of SK (over the entire spectrum of points) are summarized in Table 9.
In Table 9 a negative percentage means that ORK has smaller norms of residuals (and so has better predictions). So we see that over all three norms we get improved predictions on average with ORK. While the reduction by norm size varies greatly, it is better in this instance to use ORK as a metamodel than SK. However, this is a single problem instance with a specific set of features. We need to replicate this process many times to have confidence in the results, and so we did. In Figure 59 we show the mean norms over many replications of the base problem.

Table 9: Percentage Decrease in Norms (ORK from SK)

<table>
<thead>
<tr>
<th></th>
<th>1 Norm</th>
<th>2 Norm</th>
<th>Inf Norm</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORK Best Observation</td>
<td>+7.0%</td>
<td>+19.1%</td>
<td>+17.6%</td>
</tr>
<tr>
<td>Mean</td>
<td>+2.9%</td>
<td>+4.3%</td>
<td>+0.4%</td>
</tr>
<tr>
<td>SK Best Observation</td>
<td>-5.0%</td>
<td>-10.1%</td>
<td>-6.8%</td>
</tr>
</tbody>
</table>
In Figure 59 we see the data had smoothed out substantially and clear trends are easy to spot. First the difference between the two methods on the 1 norm is slightly favorable to SK, while the 2 norm and infinity norm both solidly favor ORK. In Table 10 we see these differences numerically.

Table 10: Percentage Decrease in Norms over Many Replications (ORK from SK)

<table>
<thead>
<tr>
<th></th>
<th>1 Norm</th>
<th>2 Norm</th>
<th>Inf Norm</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORK Best Observation</td>
<td>+0.8%</td>
<td>+9.0%</td>
<td>+9.3%</td>
</tr>
<tr>
<td>Mean</td>
<td>-1.4%</td>
<td>+4.3%</td>
<td>+6.9%</td>
</tr>
<tr>
<td>SK Best Observation</td>
<td>-4.8%</td>
<td>-1.6%</td>
<td>+4.0%</td>
</tr>
</tbody>
</table>
By comparing the data from Figure 59 horizontally we see the difference in the number of points to get equivalently good metamodels. This second kind of comparison is an excellent metric because it provides a different and interesting perspective on each method. Since we are metamodeling, one of the greatest concerns is data acquisition time/effort and so quantifying the difference in methods in this view is helpful and illuminating. However this process is made more difficult because the norm curves for single instance (like in Figure 58) are not necessarily monotonic. Hence, when making a horizontal comparison between the data sets there might be multiple choices for equivalence. By averaging the results over sufficiently many realizations of the noise, we can guarantee monotonicity and the process becomes cleaner. However, there still is the problem that the various norm values (y coordinates) do not match up with each other and so some sort of interpolation must be done. To accomplish this we fit each data series with a simple function of the form shown Equation 16.

\[ y = \beta_1 e^{-\beta_2 x} + \beta_3 e^{-\beta_4 x} \]  
Equation 16

In Figure 60 we show the fitted curves to the data. It is very good fit. Using these curves, it becomes easy to estimate the value of each method in terms of the increase or decrease in amount of data required to create metamodels with equivalently good predictions.
In Table 11 we summarize the results from Figure 60, both vertically (difference in norms over the range of the number of points) and horizontally (average extra effort required to get equivalent results between the methods). Again positive values mean that ORK is superior to SK and negative values imply the reverse. This table confirms what Figure 60 showed us, SK is slightly but noticeably better on the 1 norm while ORK is slightly but noticeably better on the 2 norm. On the Infinity norm the ORK is better than SK.
### Table 11: Percentage Decrease in Norms and Data requirements (ORK from SK)

<table>
<thead>
<tr>
<th>Norm Ratio</th>
<th>1 Norm</th>
<th>2 Norm</th>
<th>Inf Norm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Norm Ratio</td>
<td>ORK Best Observation</td>
<td>+1.0%</td>
<td>+8.8%</td>
</tr>
<tr>
<td>Mean</td>
<td>-1.4%</td>
<td>+4.3%</td>
<td>+6.9%</td>
</tr>
<tr>
<td>SK Best Observation</td>
<td>-4.8%</td>
<td>-1.2%</td>
<td>+3.6%</td>
</tr>
<tr>
<td>Simulation Effort Ratio</td>
<td>ORK Best Observation</td>
<td>+2.5%</td>
<td>+12.8%</td>
</tr>
<tr>
<td>Mean</td>
<td>-5.1%</td>
<td>+4.3%</td>
<td>+6.9%</td>
</tr>
<tr>
<td>SK Best Observation</td>
<td>-12.1%</td>
<td>-1.9%</td>
<td>+5.7%</td>
</tr>
</tbody>
</table>

#### 4.6.2 Factors Which Affect Metamodel Superiority

Now we are interested exploring the factors which affect this difference in performance between ORK and SK. In reality there are a host of factors which interact with each other make one metamodel’s predictions better or worse than another. A relatively complete list:

- The problem itself, $f(x, r)$
- Dimensionality of the problem
- Number of data points
- Design of the data points
- Noisiness of the data
- Metamodelling parameters (like covariance function and order of regression model)

There simply is no way to test all of these factors completely or in concert with each other. What we have to do is limit the scope of testing and draw reasonable inferences which
can then be added to over time. What we choose to do was restrict ourselves to a single problem form (the QN type). This problem is typical of the kinds problems in OR. We also choose to use a Latin hypercube design as it is a well-rounded choice. The modeling parameters we choose were simple, first order regression and anisotropic exponential covariance functions. We choose these because we would rather err on the side of under fitting rather than over fitting. Also with our limited amount of data more complicated methods may strip away too many degrees of freedom. The other factors listed above we varied in our experimentation.

We chose to vary the number of points as a function of the dimensionality of the problem. Varying these together is necessary because the effect of dimensionality is inextricably linked to the number of points. To control for this effect we consistently used between 4 and 24 times the number of dimensions of the problem. We feel this covers the spectrum of SM situations. Generally, having fewer than 4 times the dimensionality of the problem mean almost any model will be so strapped for information that it will be of little use. Also we feel that in situations with greater than 24 times the dimensionality of the problem we generally have so much data that SM is likely unnecessary.

Lastly, we also experimented with the level of noisiness in the data. This factor is also inextricably linked to the problem itself and the amount of “true” variation in the data. These two features form a crucial relationship in the form of the signal to noise ratio. So by holding the problems fixed we can vary the noise and then see the effect of varying this crucial factor.
4.6.3 Effect of Dimensionality of the Problem

Our first extensive study beyond the base problem was in addition to holding earlier factors constant we would hold the noise level of the data constant and vary the dimensionality. By keeping the ratio of points to dimensions fixed we hold the relative difficulty level of the metamodeling task fixed, but can see how the metamodeling techniques adapt to higher dimensional applications.

In Figure 61 we plot the percentage of data points ORK requires to create equivalent metamodels to SK. These y values are the same measure as the mean values from simulation effort ratio in Table 11 with 100 added to them. The plots are made as a function of the dimensionality of the problem. So the y value at a data point is the percentage of data that ORK needs to be equivalent to SK on that norm measure. For example there is a blue point at about (4, 102); that point signifies that ORK takes 2 percent more data to form an equivalently good metamodel as SK on the one norm. We also show a horizontal line at 100 which makes the comparisons for superiority easier. If a point is above that line then SK is superior and if a point is below that line ORK is superior.
So from Figure 61 we see that at low dimensionality the characteristic performance (SK better at the one norm and ORK better at the two and infinity norms) becomes more pronounced. We also see that as the dimensionality grows these differences becomes much less pronounced. Out at our higher dimensional examples we see that the performances are quite similar, however ORK is now superior on all three norms. These observations tell us that the contrast between methods at low dimensions is stark and the use of the metamodel should dictate methodology. However at higher dimensionality the two methods are nearly equivalent in higher dimensional situations with ORK being the better of the two.

Figure 61: Percentage of Data ORK Requires to Make Equivalent Metamodels to SK by Dimensionality of the Problem
4.6.4 Effect of Noisiness of the Data

The second major feature we experimented with was the noisiness of the data. This noisiness in the data is what motivated the development of each of these methods. That is, they both have been designed to improve the metamodeling predictions over established Kriging methods like UK when the data has stochastic noise similar to what occurs in OR situations. Again by holding the amount true variation in the data as fixed, we can vary the true factor interest the signal to noise ratio.

Figure 62: Percentage of Data ORK Requires to Make Equivalent Metamodels to SK by Noisiness of Data
Figure 62 follows the same pattern as Figure 61. However this time the variable of interest (along the x axis) is the noisiness of the data. We see that in the lower noise situations SK is markedly better on all three different norms. From that we infer that as the problems become "easier" SK's predictions improve more than those of ORK and so it becomes the better suited method. On the other hand as the problems get "harder" (more noise compared to the signal) ORK becomes more attractive than SK.

4.7 Conclusions and Extensions

In this chapter we solved the problem of finding the optimal rank for reduced rank kriging (RRK) by developing a method called omni-rank kriging (ORK). ORK sidesteps the problem of searching for optimal rank by instead averaging the predictions from all possible ranks. We have shown ORK is a superior method to RRK, particularly in light of the difficulties of finding an optimal rank. ORK has shown great promise as its predictions are of similar quality to stochastic kriging (SK). We found SK more often superior on the one norm of the residuals of predictions, while ORK more often superior on the two and infinity norms. We also explored some of the factors which effect which method is superior, finding that:

- Low dimensionality exacerbates the differences in predictions.
- High dimensionality makes the differences much less pronounced and ORK seems to be slightly better on all three norms.
- Low noise makes SK more superior on all three norms.
• High noise makes ORK more superior on all three norms.

There are many interesting avenues for further study. First is expansion of the testing of then two methods. Most imperative is trying new problems and different metamodeling parameters to see how that affects the results. We have done slight experimentation with different covariance functions and found that while the overall trends remain; moving to a Gaussian correlation function improves SK relative to ORK.

Our second extension is to get a better sense of how both SK and ORK compare to different SM methods, particularly in light of their computational times and complexity to use. In particular, how much are their predictions better than simpler methods such as regression, nearest neighbor interpolation, OK, and UK?

Lastly, we do not believe that ORK is a finished method. We believe there is room for improvement in this method which may lead to it surpassing SK convincingly. One potential improvement may be in a relaxation of using every rank to a subset of the ranks. In preliminary testing not using some of the lowest ranks consistently improves predictions. Finding a practical way to consistently use an optimal or near optimal subsets of the ranks seems useful to explore.
CHAPTER 5: CASE STUDIES

5.1 Introduction:

To help build confidence in the results from chapters two though four we decided to test the findings on two case problems. This is important for two major reasons. First, the development of the results and methods in the previous three body chapters were based on toy problems. Second, some of them were tweaks made to the methods as they became needed and results become better understood. This means that testing on independent problems is needed. Remediying those issues by using more established benchmark problems and with hypothesized results already established is critical to building confidence in those results.

The first case study problem is a discrete event simulation of a clinic. Our work on this problem will focus the experimental design results (chapter 2). However instead of simply using universal kriging we will use omni-rank kriging and test if the previous results are validated.

The second case study is a location problem. In essence it is attempting to locate ambulances in a city with Manhattan driving rules and the goal being to serve unknown calls as quickly as possible. Our use of this problem will focus on the effectiveness of ORK as a simulation metamodeling tool particularly comparing it to some baseline alternatives (chapter 4). These results will include estimates of the benefits that complex SM methods (SK and ORK) may have over simpler approaches.
5.2 PRIME Clinic

The first case study problem is an example of a clinic, which is detailed in a working paper by Taheri et. al. (Taheri, Robles, Patel, Brown, & Gellad, 2014). The basics of the simulation are that patients enter, are triaged, seen by a doctor, seen for discharge, and finally exit. We have two kinds of staff, the doctors who do the third step and nurses who do everything else. This clinic is concerned with staffing usage, wait times, and time until the last customer leaves (overtime). The main control parameters in the problem are the timing of arrivals.

5.2.1 Clinic Problem Specifics

The case study simulation was built in the SIMIO simulation language to model a VA clinic called PRIME. The PRIME clinic was recently reorganized to fully implement the Patient Aligned Care Teams approach to patient care. However there are concerns that the methods of patient handling may overload the system and cause poor performance. The simulation model was built to explore this question under variable patient loads and identify potential bottlenecks.

Data collection for various service parameters was carried out over three clinic days and done via two methods. First, for two days, patients were asked to carry a data collection form with them as they received care. The staff recorded their beginning and end times for each ask performed. Second, on the third day the staff were provided a list of the patients and
asked to time their own activities. Statistics for no-shows, process times, and arrivals time were all recorded and incorporated into the simulation model.

The flow of patient through the clinic is shown in Figure 63. This process is covered in detail in the reference paper (Taheri et al., 2014). We experimented with the model extensively to identify the most important dynamics of the model. In the end we found the model to be a reasonable representation of the real world system. Further it was accepted by the real-world stake holders in the clinic.

Figure 63: Patient Flow Through the Clinic
For our metamodeling purposes there are two main responses of interest. First is the average overtime each day. This is a function of patients’ service taking longer than average and creating blockages as well as random arrivals creating congestion. The second response is the average time in system which is a surrogate for wait time. Since the service times are i.i.d. for each run, the time in system is a good measure of wait (non-value added) time experienced by patients. These two responses vary inversely with respect to our main control, namely the arrival schedule of patients. If we have patients arrival more compressed and early then we will achieve higher utilizations and less overtime, but at the cost of greater waits for the patients since they only accrue wait time once they have arrived. We also allow control over the promptness of patients from very little to perfect promptness. We modeled the promptness phenomena by assuming patients are equally likely to be early as late (via a uniform distribution). Hence this control (promptness) has little effect on overtime since but a noticeable effect on wait times. This modeling of the promptness is not a perfect fidelity representation of reality (in reality the distribution is not symmetric); however since our use of this simulation is to test our metamodeling, we are willing to make that sacrifice in fidelity.

5.2.2 Methodology for Clinic Problem’s Use as Case Study

This case study will examine the results from chapter two (experimental design). However this problem is limited in dimensionality (only up to four). The four controls are the scheduling of patients in different ways and how much control we have over the randomness
of their arrivals. We will test all the designs from chapter 2. To judge the effectiveness of each design we will test how well they can predict the efficient frontier of objectives, overall fidelity, and alpha criteria. In Figure 64 we provide an overview of our methodology.

Figure 64: Metamodeling Process for Each Design
Following Figure 64 from top to bottom we start by creating the designs. Since the problem has 4 dimensions we decided that 40 points would be a reasonable test for each design. The amount of noise on each point was tuned to a medium level by replicating each point 100 times and then averaging the results.

Next we input each design into the simulation and run the replications. We then compile the results and find point estimates for each design point by averaging its response values over all the replications. There are two responses of interest for each replication, the average time in system for patients and the average overtime worked. These two outputs are negatively correlated with each other and demonstrate the essential tradeoff between customer experience and maximally efficient operation. By following this process we obtain the first “Collect Sim Output Rectangle” in Figure 64.

The next step is to build the metamodels. Each design’s set of data is used to build its own metamodel. So we get two metamodels for each design. However, the basic metamodeling decisions for each individual data set are constant over each metamodel. Specifically the order of regression (2nd) and the covariance function (anisotropic exponential) are the same for each metamodel. We choose to use 2nd order regression because we suspected that our simulation responses would be well approximated by quadratic surfaces. Also with 10 points per dimension we are not at risk for over fitting with no interaction terms and a simple covariance function. On the other hand, the covariance parameter values and the regression coefficients are determined individually for each metamodel.
The next step is to use those metamodels. This corresponds to the “Predict at Test Points” rectangle in Figure 64. So we created multiple large designs (FFDE and SLH). We then had each metamodel predict at each of those points. We also fed those test points into the Simulation model (Second “Run Simulations” block). Now finally we compile all the predictions and compare the quality of them to each other in light of the simulation data.

5.2.3 Clinic Problem Results, Residuals of Prediction Norms

To make those comparisons we used three metrics. First we simulated 200 points for 2000 replications, sufficient to drop the error on the mean response at each point to a very low value. We then also used the metamodels to predict the responses at those same points. Doing so gave estimates of the norms of the residuals of the metamodel predictions. The norms are summarized in Figure 65. In the Figure 65 we rescale the norm values to the \([0, 1]\) interval. The FFDC design is excluded because its performance is vastly inferior to all the others and removing it allows for us to show the other nine designs performances more clearly. We see three designs are noticeably better than the others: FDE, CCDF, and NTM.
Another important question we now can answer is “how much value does the metamodel provide?” While we would like our metamodel to be a perfectly accurate and unbiased predictor of the mean response value, we know that there will still be some error in those predictions. In particular, we would like to know how much simulation effort would be required to get an equivalently good prediction through replications in the full model compared to using the metamodel.

The cost to predict at a new point via the metamodel is negligible relative to the cost of simulation effort and so we only focus on the cost of simulation effort. This assumption is completely justifiable if simulation metamodelling is appropriate. This is because in most
applications the fitting of the simulation data and then predicting every point the metamodel will eventually be used at costs less that the cost of simulating a single new point.

We solve for this equivalency empirically by taking the MAE and MSE from above and find how many replications are required from the full simulation to achieve similar MAE and MSE. To carry this process out we utilized the 2000 replications for each data point. We sampled subsets of those replications and found the mean response for each subset. Then, we estimated the error by repeating this process for many different subsets. By varying the size of the subset we were able determine the proper level of error (MAE and MSE) that is equivalent to our metamodel. In Table 12 we summarize the results.

Table 12: Replications Required for Brute Force Simulation to Be Equivalent to Metamodel

<table>
<thead>
<tr>
<th>Response</th>
<th>MAE</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response 1 (Overtime)</td>
<td>113</td>
<td>123</td>
</tr>
<tr>
<td>Response 2 (Time in System)</td>
<td>140</td>
<td>152</td>
</tr>
</tbody>
</table>

Table 12 shows that it would take around 120 or 145 replications respectively by response to get an equivalently good prediction for a new point compared to a prediction from our metamodel. Our metamodel was built on 4,000 total replications (40 points w/100 reps). The values in the table are for each point to be predicted. So if we wish to estimate the response at 500 new points, we would need about 60k-72k total replications to get equivalent quality that comes from the metamodel. So clearly metamodeling can pay dividends depending on the circumstances. In fact, there is reason to believe that with more skillful
allocation of the initial simulation budget, a better metamodel could have been produced. This better allocation could be done via sequential design or better point/replication allocation.

5.2.4 Clinic Problem Results, Alpha Criteria

Second, we find the alpha criteria for each design’s metamodel by randomly picking weights for the two criteria to make a composite objective. We then perform the standard process in determining the alpha criteria outlined in Chapter 2. We repeatedly sample possible weights to get more robust alpha criteria estimates. For these estimates we include some additional design points which focus on potential extreme values (each point also with 2000 replications). In Figure 66 we show the alpha criteria values for each design at three levels ($\alpha = .01, .05, .25$). We see that the FFDE, CCDF and CHD seem to be most consistently best for optimization.
5.2.5 Clinic Problem Results, Efficient Frontiers

Lastly we explore which design does the best job of predicting the possible efficient frontier between the two objectives. To determine the frontier, we formed each metamodel with a very large number of points (2000) and then plotted their responses. Then we found the efficient frontiers which are feasible according to each metamodel. We then plotted those against the efficient frontier possible from the extensive simulation data we had collected. In Figure 67 we plot the efficient frontiers on the blue to red spectrum and the direct simulation data in green.
We can clearly see that many of the efficient frontiers are not particularly good. In Figure 68 we exclude the clearly inferior designs and plot the remaining ones. From inspection it appears that the CCDF is the best design. This gives us good evidence that this metamodel does the best job of sensitivity analysis, particularly where it matters most the prediction of optimal solutions.

Figure 67: Metamodel Efficient Frontiers Compared to Data from Full Simulation
So from all these metrics we concluded that the best designs for this case example are the CCDF, FFDE, and NTM. There are other designs which may be competitive on one metric or another; however the three designs just specified are consistently excellent. So to the question which needs to be answered: “Do these results agree with the results from the earlier chapters?”

Figure 68: Subset of Metamodel Efficient Frontiers vs Data from Full Simulation
5.2.6 Clinic Problem Results, Agreement with Earlier Results

Overall we find that this case study results agree with the prior results. First the basic result that using the right tool for a job is critical is fully confirmed. Since we choose to use second order regression model we would expect designs which are well crafted towards that end do well. We see that to be true, since the CCDs become much better relative to our results from chapter 2. Second, we actually repeated the process outlined above using a 1st order regression model. Upon this repetition see that he CCDs become less useful and the LHs become relatively much better.

The results with regards to fidelity versus optimization also conform to the earlier chapter. We see this first in the residuals of the prediction norms. Picking designs which are well suited to the second order regression create metamodels which have good fidelity throughout the whole space. Second looking at the suitability for optimization we see the best design for metamodels are those which spread the data to edges of the design space (FFDE, CCDF, CHD, and to lesser extents NTM and MMLH). By doing that, they get good estimates for the regression parameters and have points nearby for when the optimal values are located at the edges of the design space.

5.3 Ambulance Location

The second case study is a location problem. The details are given in a paper on simulation testbed problems (Pasupathy & Henderson, 2006) and freely available (with
computer code) from simopt.org. In the problem we attempt to best locate ambulances in a city with Manhattan driving distances and the goal being to serve unknown calls arising in the space as quickly as possible. For our problem we use the default parameter values given in the aforementioned paper.

5.3.1 Ambulance Location Problem Specifics

In this problem we are tasked with placing a specified number of “ambulance bases” in a 2D space. The space for this problem is assumed to be a unit square. The calls arrive according to a specified density function; which for our problem, we take the one recommended in the paper. Our objective is to place the bases to minimize the wait that service calls will experience.

When a call is registered, its service is assigned to the closest free ambulance. If all ambulances are busy, the call is placed into a queue for service and then served in FIFO order. Once a call is assigned, its ambulance travels towards it at a constant rate, using Manhattan travel (no diagonal travel, on N/S and E/W). After serving a call, ambulances return to their bases at a return speed, unless they have a call they need to respond to waiting on them.

We are interested in how to best place the given number of ambulances bases to minimize the long run average response time. We will solve this problem for different numbers of bases (and so different dimensionality of the problem). For simplicity we will
vary the arrival rate as a function of the number of ambulances such that the service load remains at the same level. We will be able to achieve better response times with higher numbers of ambulances due to lower travel distance and pooling of servers. This variation is not a concern because we will not be comparing response times across different levels of the number of bases.

5.3.2 Testing Methodology

We will test the usefulness of Omni-Rank Kriging and Stochastic Kriging as surrogates for optimization. We will compare their usefulness to two other methodologies: Brute Force Simulation and a Genetic Algorithm. Since there is no known optimal solution to this problem, we will judge the performance of each method by looking at how much they can improve the objective value within a limited computational budget.

In general, this kind of optimization problem often has a human decision marker as a key component. Inclusion of a human decision maker is feasible because data acquisition is so expensive and slow that having a human make judgments in real time about where to spend computational budget can vastly improve results without meaningfully slowing down analysis. However, we do not want to let human judgment bias the results so we will instead try to set up effective and flexible frameworks which each methodology can fall back on to approximate human judgment.
The Brute Force (BF) approach is by far the simplest. It simply tries placing more and more points in the space. Whenever we find a point is better than any so far, BF changes its estimate of the optimal to the new one. It continues generating new points until its budget is exhausted. The Genetic Algorithm (GA) is a well understood methodology which we think has good potential for this problem. It is in many ways an intelligent BF approach which tries to expand on solutions which are promising when it comes to generating new points. Since this problem is tremendously multimodal, it is well suited to using a GA since GAs are resistant to getting caught in local optima. A GA will focus on promising areas better than the BF approach. However, compared to other optimizer methods, the GA is very reliant on function evaluations, which makes it rarely effective for simulation metamodeling by itself.

We use both ORK and SK in direct calculation for estimating good points. To do this we start with a set of data points. We then take that data and metamodel it using each technique. Once the metamodels are built, we predict the response value at many points through the entire space. We then take a small subset of the points predicted to be the best by each metamodel and do local searches using the metamodel for local optimization cheaply. We then select the best point discovered and use it as an estimate of the optimal. If there are no points discovered better than the best data value, we use the best data value instead. There many potential ways to improve these methods (sequential designs, saving some simulation budget for distinguishing between candidates, etc.) but we decided to keep it simple and see how the results turn out.
We will test these methods for solving the ambulance location problem with 3, 5, and 8 ambulances. The values of solutions will improve as the dimensionality increases. Also, the difficulty of the problem will increase due to the fact that we will not be changing the computational budget. This will allow us a bit more perspective on each method. In the lower dimensional examples there should be sufficient data that methods get close to the optimal value because they have plenty of data at their disposal. On the other hand, to be successful at the higher dimensional examples methods we will need to not get caught searching local maxima and try to ensure good global coverage with their very limited data. Ultimately, we tested these methods 60 times using different data sets to get confidence in our results.

5.3.3 Case Study Results

In Figure 69 we show the results for the three ambulance problem. Along the x axis are function evaluations and along the y axis are the objective values at that point. The GA, ORK, and SK models do not have values at the left most part of the x axis because they require a minimum amount of data to run. However we can say that those methods have the same value as BF in that space. This is because they have not yet begun their processes and they would simply be the same as the BF approach, given the same data. This is in fact what we did. We used the same data from the BF for each other method where possible as a variance reduction technique.
In Figure 69 we see the steep drop in BF for the first 40 data points (from 1 to 40) and then slow but steady progress. The GA, which starts at 20 points, follows a very similar pace to the BF but is slightly better. This is as expected since the GA is in essence an improved version of the BF. At 30 data points in the budget the ORK and SK curves begin. We see they are substantially lower than both the BF and the GA. This is an emphatic demonstration of the potential value of SM using these methods. In Figure 70 we zoom in on this lower right region so we can see greater detail.
In Figure 70 we zoom in on the performance of ORK and SK. They both start vastly better than BF or GA. In fact using as little as 30 data points with an ORK or SK metamodel provides for better estimation than BF or GA with 200. Comparing ORK to SK, we see they perform at a very similar level to each other. ORK gives us slightly better optimal solutions than SK; however the differences are not practically significant.
In Figure 71 we see the performance of all four methods on a five ambulance problem. Again we see that BF is the worst performer. From 20 points to 200 it improves the objective value from about 0.246 to 0.236. The GA starts at .246 as well but it is able to improve the objective value to 0.235. The real strength of the surrogate methods is their ability to make good models from limited data. That is why they start with such low objective values. On this example they take the objective value from around 0.235 at 60 data points.

Figure 71: Performance of Optimizer Methods, 5 Ambulance (10 Dim) Problem
points to .2315 at 200 points. Again the differences between ORK and SK is to not practically significant.

![Graph](image.png)

**Figure 72: Performance of Optimizer Methods, 8 Ambulance (16 Dim) Problem**

In Figure 72 we see the same graph, but this time on an 8 ambulance problem. The story is the same, BF is worst, GA is between BF and the ORK/SK methods, and SK and ORK are essentially the same. What we notice comparing this figure to the earlier figures is that the surrogate model’s performance improves over the amount of data more substantially. This occurs because for the surrogates to be useful they must reach a reasonable fidelity to
the full simulation. And to get a good model in higher dimension requires more data than in
the lower dimensional examples.

So in this case study we have shown that the SK and ORK provided substantial
benefits to optimization of a difficult simulation problem. We found that using these
surrogates can reduce the data required by a factor of 5 to 10 depending on the situation. This
demonstrates the value that surrogate modeling provides. With respect to the competition
between ORK and SK we find that there is no practical significance between their
performances. The way we used ORK and SK as surrogates may not have been maximally
powerful, so possibly improving that would make the differences between the two methods
more clear. Also enhancing the surrogate methodologies would likely make their superiority
to the other methods more substantial.
CHAPTER 6: CONCLUSIONS

In this dissertation we have made three noteworthy contributions to the literature on simulation metamodeling. They are:

- An analysis of what comprises good experimental designs for practical (non-toy) problems when a kriging metamodel is used for analysis;
- An explanation and demonstration of how reduced rank decompositions can improve the performance of kriging as a tool for metamodeling;
- Development of a methodology, omni-rank kriging, which solves the problem of finding the optimal rank for use in reduced rank kriging.

The first body chapter of this dissertation focused on how different experimental designs led to better or worse simulation metamodels. We choose to use kriging as our metamodeling method because it is consistently good at a variety of problems. This is because it is flexible enough to fit many problems but still powerful due to its complexity and solid underlying theory. Our study filled a gap in the literature by looking at higher dimensional issues than are commonly examined in geostatistics. Further, we use a wider variety of designs than are often seen in the OR literature.

Ultimately, we found confirmation of basic intuitions. First, we found that picking the right design is dependent not only on the features of the problems but also the purpose of the simulation metamodel. In particular, looking at where difficult to model features may be in the design space is valuable. Also trying to place data points near where the optimal values
are likely to be located makes for a much more effective metamodel for surrogate optimization.

The second body chapter looked at how using a singular value decomposition and then a reduced singular value decomposition can improve kriging predictions. We showed through careful tracking how this change propagates through kriging empirically to artificially smooth the kriging weights. We called this smoothing of the weights “democratization” because instead of just a few points having large effects on the kriging prediction, using the RSVD causes many points to have small influence on the prediction.

Lastly, we showed that this methodology could outperform standard universal kriging even with a nugget effect on limited examples. This was ultimately because the data from an operations research simulation does not fulfil the criteria to make kriging the optimal predictor, and reduced rank kriging compensates for this better than universal kriging. We found that reduced rank kriging was normally outperformed by stochastic kriging but the differences were not large and so reduced rank kriging showed great overall promise.

The third body chapter focused on the critical open question from the second body chapter which was “how to find the right rank to use in reduced rank kriging?” Our initial approach was based on cross validation, where by leaving out a subset of the data we could estimate the quality of a metamodel using a certain rank by using the excluded data as a check. This however proved to be unsuccessful because of insufficient data and noisiness of the data, the very things which motivate most simulation metamodeling to begin with. So we
worked around the problem by instead averaging the results for every possible rank and called the result omni-rank kriging

We then experimented with omni-rank kriging and found it to be not only vastly superior to cross validation approaches but even superior to reduced rank kriging with full information. So the next step was to compare it to a peer method, stochastic kriging. We found it to be very competitive with stochastic kriging. Stochastic kriging was generally better with regards to absolute errors of prediction, and omni-rank kriging was better in regards to squared errors of prediction.

We also tested these results on two case study problems. We used the first problem (a discrete event simulation of a clinic) to test our conclusions regarding experimental design. We used the second (a simulation of a location problem) to test our results regarding omni-rank kriging. We found general agreement in both cases. On the first case study problem, we found further evidence that effective planning of experimental designs should consider both the problem characteristics and the eventual use of the metamodel. We also showed on this example how it is possible to make metamodels which, after a reasonable up-front cost, can provide large computational savings with repeated use. On the second problem we saw that omni-rank kriging and stochastic kriging are both valid surrogates for optimization. Both methods markedly speed up optimization over more brute force methods.

There is one final thought which this dissertation has not explicitly sought to confirm through all the results, however is an unavoidable conclusion. That is a great faith in the power and robustness of the central limit theorem. This has been caused by repeated
examples of how combining many predictors into one can make for excellent predictions. This is even true if the individual predictors are mediocre and potentially dependent on each other. This is a parallel to the concept of the wisdom of crowds. Under weak and easy to fulfil assumptions regarding the predictors, the result of combining many mediocre predictors can and often does result in an acceptable to excellent prediction. We believe that current literature spends too much effort on seeking the best individual predictor methodology. We think instead it should focus more on how different predictor methodologies can be mixed and combined to create composites which outperform even the best single methodology. We believe doing this will lead to better overall predications.
REFERENCES


Martin, J. D., & Simpson, T. W. *A study on the use of kriging models to approximate deterministic computer models.*


APPENDIX A: BRIEF LITERATURE REVIEW ON EXPERIMENTAL DESIGN

This Appendix gives a short literature review on experimental design from statistics. This information should shed further light on how some of the designs which were considered.

The first subset of literature in statistics we must consider is that which seeks to create optimum design of experiments (frequently abbreviated to o.d.e.) (Atkinson, 2007; Fedorov, 1972; Jack Kiefer, 1985). In this branch of literature the goal is to maximize the power of the statistical test employed or maximize the accuracy of our parameter estimators. This approach is intuitively appealing to Operation Researchers since it employs familiar optimization concepts. However, these designs suffer from two problems which make them less suitable for use with Kriging.

The first problem that o.d.e. designs suffer from is that they are trying to optimize a single criterion. Thus even for the smallest gain in our objective we will lose other beneficial properties (aside from specified constraints). Box and Draper (GEORGE E. P. BOX & DRAPER, 1975) give 14 potential goals in experimental design and say how focusing on one to the detriment of the others seems potentially unwise. So while o.d.e. will lead to designs which are optimal in their specified objective, they give up much more than is good practice in other potential objectives (George E. P. Box & Draper, 1987).
This first problem then leads to our second concern: the robustness of these designs in the face of misspecification. There has been substantial research which shows the general underperformance of regression (RSM) as simulation metamodeling tool (Ankenman et al., 2010; J. P. C. Kleijnen & W. C. M. van Beers, 2005; Reis dos Santos & Porta Nova, 2006; Villa-Vialaneix et al., 2012). In most papers, regression is the baseline or strawman that is defeated handily by every other method tested. So we do not believe that some kind of regression model is likely to explain our data effectively. o.d.e. methods rely on the assumption of perfect knowledge of the form of the response and may be highly sensitive to this assumption being false (G.E.P. Box & Draper, 2007). This assertion means that one of the crucial assumptions for these o.d.e. designs seems unlikely to be true, and therefore likely to result in poor outcomes (G. E. P. Box & Draper, 1959; Müller, 2007).

While these problems are acknowledged, we still will include some of these designs for comparison with (hopefully) more appropriate methods. The first such design is the Fractional Factorial Design (FFD) (Giesbrecht & Gumpertz, 2011). This design is a basis for many other designs. It allows the researcher to make inferences, even in high dimensional situations, without an exponentially increasing the number of observations. The second design is the Central Composite Design (CCD) (Giesbrecht & Gumpertz, 2011). This design augments a FFD by adding points in the middle of the design space, as well as on the centers of the design space’s faces (called axial points). Doing this requires additional points on the order of $d$, and so is a reasonably small increase over the FFD. Adding these points allows for good sensing of curvature in the RSM. Two other potential designs are Box-Behnke and
Box-Behnke Designs requires on the order of the dimension squared points. This number is too many for high dimensional simulation metamodeling, and so will not be pursued here any further. Lastly from the classical design literature are the “alphabetic” designs. Each of these seeks to maximize some virtue or idea about designs. While there is a multitude of these (Han & Chaloner, 2003; Hardin & Sloane, 1993; J. Kiefer & Wolfowitz, 1960; Pukelsheim & Studden, 1993; Toby, 2000), we will not pursue them any further since their assumptions are almost surely not satisfied by our problems and often their results are sensitive to deviations from their assumptions.

The second subset of research can be broadly classified as studying the effects lattice designs on Kriging’s predictions. These designs are quite simple, a repeating pattern in the experimental space. The most common lattice designs are grids. The three most common grids in literature are square, triangular, and hexagonal, with square being the most common among those. There have been a few papers which tried to distinguish which grid shape was best, but they generally show that the differences are slight, with different papers finding competing results (Olea, 1984; Yfantis et al., 1987). Also, it is important to note that these papers only study a two dimensional $\mathcal{X}$, so their applicability to our situations is likely limited.

The strength of these grid designs is that they cover the space of $\mathcal{X}$ (they keep points well away from each other). However, these designs are vulnerable to the curse of dimensionality and are not receptive to augmentation, both of which are problems for analysts (K. Fang, 1994). Finally, there are no good high dimensional representations of non-
rectangular grids. It is for this reason we will proceed with only rectangular grids in our study.

In high dimensional situations, the grid designs become infeasible due to the curse of dimensionality. So we choose to link the Fractional Factorial Designs and the grid designs to cover the data poor and data rich situations respectively. This decision is sensible since grid designs are another name for Full Factorial Designs.

Beyond the simple grid designs, but still in the category of lattices, are a wide range of designs called Number-Theoretic Methods (NTM) introduced by Fang (K. Fang, 1994). These methods attempt to insure good dispersion of experimental sites by minimizing the discrepancy metric, also known as the U-criterion. They also are built to be effective in higher dimensional problems so we will include them in future work.

These, and other, lattice type designs generally are intended to remedy the problems having large “holes” (areas without a nearby data point) in the design space. There is another subset of the literature which shares this desire the minimize holes and/or maximize space between data points, but drops the requirement of lattice structure. Designs motivated by these goals are called “Space Filling Designs.” There are a variety of particular methods to constructing a space filling design, but they generally either try to minimize the size of the holes or maximize the distances between data points.

There are a few ways to attempt to maximize the “repulsion” between points. Johnson et al. (Johnson, Moore, & Ylvisaker, 1990) give an illustrative article where they show how
slight differences in the exact requirements vary the resultant designs. In some idealized cases these methods will reduce to a grid design. However, if the design space is oddly shaped or the number of points is not just right, this design will not apply. To get this maximum repulsion, designs maximize the distance to the closest neighbor for each point. This criterion is a very difficult optimization problem having dimension equal to \( d \times n \) (number of dimensions multiplied by number of design points). What is sometimes done instead of trying to solve this problem all at once, is to add points sequentially so that the effort to add each point is not so severe, and then all the points can be placed without too much effort. This idea is implemented in Coffee-House Designs (CHD) which are shown in (Müller, 2007). CHD seem to have a number of properties which should give robust designs, and their reasonable computational demands mean they are a design we will include future testing.

Latin Hypercube Designs (LHD) are popular in a wide array of fields and are either the first or second most commonly used designs in SM (rectangular girds are the other). LHD are a version of stratified sampling, which is also used in other simulation contexts. LHD splits each dimension of \( \mathcal{X} \) into \( n \) slices, resulting in \( \mathcal{X} \) being split into \( n^d \) sub-hypercubes. Then points are placed such that there is one, and only, point in each slice for each dimension. The most basic version, Latin Hypercube Sampling (RLH), was proposed by Mckay et al. (McKay, Beckman, & Conover, 1979) and it picks the sub-hypercubes randomly, and then places the design points randomly inside of each sub-hypercube. There have been a great variety of modifications to LHD. One of the most popular modifications
intends to balance the placement of design points through the design space. Both Owen (A. Owen, 1994; A. B. Owen, 1992) and Tang (Tang, 1993) independently using randomized orthogonal arrays (OA). By using orthogonal arrays the process is stratified further, guaranteeing better covering than LHS. The difficulty with this method is in creating the orthogonal arrays, which will be expanded on later. Another modification is making the LHD symmetric. Park (J. S. Park, 1994) and Morris and Mitchell (Morris & Mitchell, 1995) found that when looking for optimal LHDs they often exhibited symmetric properties. Ye (Ye, 1998) gives an implementable method for constructing Orthogonal Column Latin Hypercube Designs (OCLHD) which is probably slightly outside of the design skills of an OR analyst. Lastly, there have been attempts to optimize LHD for some criteria (o.d.e. within the LHS setting). The criteria which will be tested in this dissertation will be” maximin”, which tries to maximize the minimum distance between points (like in the CHD).

There are a variety of other space filling methods. Hammeresley sequences were created as another way to effectively cover the design space (Hammersley, 1960). They also have the benefit of being computationally cheap to make (like RLH). Kalagnanam and Diwekar (Kalagnanam & Diwekar, 1997) show how using Hammeresley designs has many useful properties. Another interesting technique to use is maximum entropy methods (Currin, Mitchell, Morris, & Ylvisaker, 1991). However both these methods have not gotten much use outside of Monte Carlo Integration, and so we will not use them for designs.

Orthogonal Arrays are a design technique with tremendous flexibility; they can create Full Factorial, Fractional Factorial, and many kinds of mixed designs. OAs, and even the
entire field of DOE, are sometimes mistakenly attributed to Taguchi (see Ryan (Ryan, 2007) for details). Taguchi did not create the idea of OAs, however he did popularize them as a concept for the design of experiments.

Due to their flexibility, to pick out a single OA to represent all OAs would be similar to picking only a single space filling design to stand for all the various space filling methods, or d-optimal designs to stand for all classical designs. OAs deserve their own study of what makes for good experimental designs for SM. Since there is no established knowledge on what kinds of OAs to use for SM, and coding a flexible OA method would be a significant problem in its own right, we will leave that as an open topic and not use them for this study. Further, there is no single, unified method for constructing OAs. Hedayat et al. (Hedayat, Sloane, & Stufken, 1999) list more than a dozen separate methods for construction OAs of various sizes. This diversity means that adding them into this study is no trivial task and will be left for research beyond this dissertation.

There are a wide variety of other possible designs from the statistics community. However, since this dissertation is not focused on reviewing past designs, but instead focused on testing designs, we will not continue to list other possibilities. The two introductory texts listed at the beginning of this section are good references to lead interested readers into wide array of other design options.
APPENDIX B: ILLUSTRATIONS OF THE DESIGNS FROM

CHAPTER 2

This Appendix gives illustrations of each design from Chapter 2. These illustrations are given to help make details of the designs easier to understand. These details also should give the reader more knowledge about how various designs resemble each other. Figure 73 shows one realization of 20 random points in two dimensions. It shows how this design often suffers from large unsampled holes like the one in the lower left quadrant.

Figure 73: Simple Random Sampling, 20 Points, 2 Dimensions
Figure 74 shows another realization, this time with 80 points in two dimensions.

Figure 74: Simple Random Sampling, 80 Points, 2 Dimensions
Figure 75 gives a realization with 80 points in three dimensions.

Figure 75: Simple Random Sampling, 80 Points, 3 Dimensions
Figure 76 shows a factorial design, with points on the edges of the design space. It has 80 points in two dimensions. There is an 8 by 8 grid which makes up 64 of its points. The other 16 points are placed at random by a simple random sample. The use of random points to complete the design (from 64 grid points) is so that the design can be run with any number of points in any dimension. This decision is covered in the methods section of Chapter 2.
Figure 77 is a factorial design of 64 points in three dimensions. It shows how this design works without any random points.
Figure 78 shows an example of a Fractional Factorial Design. This separation of points is what happens to the Factorial Design when it has less than $2^d$ points. To make up the gap between the required number of points and a Fractional Factorial Design of maximum allowable resolution, we add other points on the hypercube not already included in the design. For instance in this example the Fractional Factorial Design is:
{(0,0,1), (0,1,0), (1,0,0), (1,1,1)}. To this start we added {(1,0,1), (0,0,0)}.
Figure 79 shows the same factorial design as in Figure 76, but with the design points coming from the center of their grid spots instead of the outermost corners. As can be seen by comparing the Figure 79 with Figure 76, the randomly placed points are not subjected to this centering.

Figure 79: Centered Factorial Design, 80 Points, 2 Dimensions
Figure 80 shows an Inscribed Central Composite Design with 23 points in two dimensions. The difference between this design (inscribed) and the Central Composite Design which is faced is the spread of the core points. In an inscribed design, the core points are scaled into the center of the design space based on the dimensionality of the problem. (See Figure 82 for comparison)

![Central Composite Design, 23 points in two dimensions](image)

**Figure 80:** Central Composite Design, 23 Points, 2 Dimensions
Figure 81 shows an Inscribed Central Composite Design with 15 points in three dimensions. From this we can easily see the three kinds of points. There are eight core points which are a $2^3$ design, the center point and the six axial points.

Figure 81: Central Composite Design, 15 Points, 3 Dimensions
Figure 82 shows a Faced Central Composite Design with 23 points in 2 dimensions. In this faced design, the core points are allowed to spread all the way out to edges of the design space. (Compare to Figure 80)
Figure 83 shows a Number Theoretic Design with 25 points in two dimensions. This design does a good job of spacing points throughout the space.
Figure 84 shows a Number Theoretic Design with 80 points in two dimensions. We see patterning beginning to really show with the increase in points over Figure 83. This patterning will only increase with the addition of more points (this patterning is similar to that of modulus-based random number generators). This design looks weak in lower dimensions, but that is partly to be expected since it is designed to perform best in high dimension.

Figure 84: Number Theoretic Design, 80 Points, 2 Dimensions
Figure 85 and Figure 86 show the Number Theoretic Design with 80 points in three dimensions. Figure 85 shows how from certain angles it may appear that the design project very poorly (there are large unsampled holes). Figure 86 shows how from other angles these holes are not as apparent.

Figure 85: Number Theoretic Design, 80 Points, 3 Dimensions
Figure 86: Number Theoretic Design, 80 Points, 3 Dimensions
Figure 87 shows a realization of the Coffee House Design with 20 points in two dimensions. It is important to remember that the Coffee House Design relies on an amount of pseudo-random numbers to be created. Hence each realization will be distinct (up to some accuracy).

Figure 87: Coffee House Design, 20 Points, 2 Dimensions
Unlike realizations of the Simple Random, sampling the variation from realization to realization is substantially less in Coffee House Designs. This variation can be seen by looking at the small amount of variation in design from Figure 87 to Figure 88.

Figure 88: Coffee House Design, 20 Points, 2 Dimensions
Figure 89 shows a Coffee House Design with 80 points in three dimensions. It may be tough to tell with only a single perspective, but the points are quite effectively spaced out (well repulsed) through the space. To further illustrate this point, the minimum distance between any two points is 0.2137. For comparison, a randomly sampled set of points would have an expected value of 0.03837 ± .0001 for its minimum nearest neighbor distance. On the other hand, an optimally spaced design (maximizing the minimum neighbor distance) is a grid design. This grid design achieves a minimum neighbor distance of 0.3333. So while there is some gap in terms of being fully spaced out, this design does achieve a good spacing.

Figure 89: Coffee House Design, 64 Points, 3 Dimensions
Figure 90 shows a Random Latin Hypercube design with 20 points in two dimensions. This design achieves a good mix of spacing points and clustering them such that we get information about data points close together and far away. This is helpful when we try to build a variogram.
Figure 91 shows another realization of the Random Latin Hypercube Design with 20 points in two dimensions. In this realization we see better coverage in the top right corner.
Figure 92 shows a realization of the Random Latin Hypercube Design with 80 points in three dimensions. Since each dimension is independent, each of the 2-d projections of this design look like any other 2-d realizations of the design. As expected, the design looks very good when projected down into 1-d in each of the three dimensions.
Figure 93 shows a realization of a Maximin Latin Hyper Cube Design with 20 point in two dimensions. Again it is important to stress that this is only one realization since the method for creation relies on a RLH design to initialize it. It then does a row exchange algorithm to improve the design on the criteria of interest (maximizing the minimum distance between points).

Figure 93: Maxmin Latin Hypercube Design, 20 Points, 2 Dimensions
Figure 94 shows a realization of a Maximin Latin Hypercube Design with 80 points in two dimensions.

Figure 94: Maxmin Latin Hypercube Design, 80 Points, 3 Dimensions
Figure 95 shows a realization of a Symmetric Latin Hypercube Design with 20 points in two dimensions. Note how every point has matching point opposite from itself through the point (.5, .5).
Figure 96 shows a realization of a Symmetric Latin Hypercube Design with 21 points in two dimensions. Note how the point (.5, .5) is now present in the design. Also the matching of each point with its opposite can still be done.
Figure 97 shows a realization of a Symmetric Latin Hypercube Design with 80 points in three dimensions. It is hard to make out the symmetry in the design from this view. However on each of the two dimensional projections the symmetry is much easier to make out, exactly as in figures Figure 95 and Figure 96.

Figure 97: Symmetric Latin Hypercube Design, 80 Points, 3 Dimensions
APPENDIX C: TESTBED PROBLEM FORMULATIONS AND ILLUSTRATIONS

C.1 Slope

Slope (SL) is a hyper-plane in \( d \) dimensions. It a simple linear function in each variable. It is totally separable, has no curvature, and it has its optimal (maximal) value at a vertex of \( \mathcal{X} \).

\[
f(x) = \beta^T x
\]

Where for a given problem instance:

\[
\beta = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_d \end{bmatrix} \text{ and } \beta_i \text{ are i.i.d. } \sim \text{Uniform}[-2, 2]
\]

Figure 98 shows two realizations of SL.
Figure 98: Visualizations of the Slope Problem
C.2 Hyper-J

Hyper-J (HJ) draws its name from what it’s two-dimensional cross sections look like, J's. The prefix, “hyper,” was added to emphasize the existence of the problem in $d$ dimensions. This problem is separable, but has significant curvature. It has explosive growth on one edge. This is the “difficult to model edge behavior” which is studied extensively in the results section. It has its optimal (minimal) value at a vertex. Its combination of difficult to model edge behavior and placement of the optimal value at a vertex makes it a challenging problem to metamodel.

$$f(x) = \sum_{i=1}^{d} \left( \beta_i y_i^2 + \frac{y_i}{1 - y_i} \right)$$

Where for a given problem instance:

$$w_i = \text{Bern}(0.5); \ y_i = \begin{cases} 0.96(1 - x_i) + 0.02 & \text{if } w_i = 0 \\ 0.96(x_i) + 0.02 & \text{if } w_i = 1 \end{cases}$$

$$\beta = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_d \end{bmatrix} \text{ and } \beta_i \text{ are i.i.d. } \sim \text{Uniform}[0, 2]$$

Figure 99 shows two realizations of HJ, with the negative transform applied (so the optimal value is the maximum)
Figure 99: Visualizations of the Hyper-J Problem
C.3 Newsstand

Newsstand (NS) problem is based on a “newspaper boy” problem, but extended into many dimensions with product substitution. The product substitution is done by creating product groups which substitution inside of would be allowed. When we are out of one item in a group we can substitute another item from the same group at a small penalty. The real world example of this might be someone looking to a USA Today might settle for a local paper instead or someone looking for a Sports Illustrated might accept an ESPN the Magazine. The Newsstand problem is smooth and quadratic in shape without any explosive growth on edges. The objective on this problem is to maximize expected profit by stocking the proper products. For simplicity we allow fractional stocking and demand. The formulation for this problem is given on the next page.
\[ f(x) = E \left[ \sum_{i=1}^{N} \sum_{j=1}^{n_i} sales_{ij} - costs_{ij} \right] \]

Where for a given problem instance:

\[ N = \text{number of major product types} \]
\[ n_i = \text{number of minor product types in major type } i, \quad i = 1: N \]
\[ \sum_{i=1}^{N} n_i = d \text{ (dimension of the problem)} \]
\[ c_{ij} = \text{Cost to buy a product } ij \sim U[.5, 1] \]
\[ p_{ij} = \text{Sales price for product } ij \sim U[1.5, 2] \]
\[ \text{minimum demand} = 0 \]
\[ b_{ij} = \text{maximum demand for product } ij \sim [5, 10] \]
\[ \beta_i = \text{Sales price for satisfying demand for } ij \text{ with another prod. from type } i = 1.00 \]
\[ \gamma_i = \text{penalty for not satisfying demand for product } ij \text{ with anything} = -.5 \]

Figure 100 shows two realizations of NS.
Figure 100: Visualizations of the Newsstand Problem
C.4 Queueing Network

Queueing Network (QN) is the problem of finding how to set service rates at different nodes in a queuing network. The objective is to minimize costs, which are made up of two components: waiting costs and server costs. The waiting costs force us to serve “customer” (process orders, machine parts, etc.) at reasonable pace. The server costs mean that we can’t increase the service rate needlessly. As with the Hyper-J problem we will impose a negative transform to turn this minimization problem into a maximization one.

There are two factors making this problem difficult to metamodel. First, the optimum often occurs very close to the explosive growth edges of the design space. This is tough for any metamodel to deal with since small deviations in data feed into the model can change the metamodel substantially. Similarly, when testing the metamodel small deviations in the points which are predicted at can substantially change the quality of the prediction from the metamodel.

The second difficulty is that the problem’s behavior changes radically in different areas of $\mathcal{X}$. The problem is quadratic looking near the optimum, looks linear in some other areas, and has explosive growth near one edge of each dimension (resultant from insufficient service at a node and so an unstable system). This means that the behavior around the optimum is not consistent in opposite direction over equal distances. (This is radically different from NS).
Figure 101: Visualizations of the Queuing Network Problem