DATA-BASED MATHEMATICAL MODELING: DEVELOPMENT AND APPLICATION

or

HOW TO BUILD A MAPPING NEURAL NETWORK

By
Mahmoud-Reza Banan
and
Keith D. Hjelmstad

A report to sponsors:
The Army Research Office
The National Science Foundation

Department of Civil Engineering
University of Illinois at Urbana—Champaign
Urbana, Illinois

April 1994
Data-Based Mathematical Modeling: Development and Application

or

How to Build a Mapping Neural Network

Mahmoud-Reza Banan
and
Keith D. Hjelmstad

Department of Civil Engineering
University of Illinois at Urbana-Champaign
Urbana, Illinois USA

A report to sponsors:
The Army Research Office
The National Science Foundation

April 1994
Mahmoud-Reza Banan

and

Keith D. Hjelmstad

Department of Civil Engineering
University of Illinois at Urbana-Champaign
205 N. Mathews
Urbana, Illinois 61801
U.S.A.

Data-Based Mathematical Modeling:
Development and Application

or

How to Build a Mapping Neural Network

© 1994 by M.-R. Banan and K. D. Hjelmstad

All rights reserved. No part of this document may be translated or reproduced in any form without the written permission of the authors.

Printed in the United States of America.

9 8 7 6 5 4 3 2 1
This report presents a general method for developing data-based mathematical models for complex problems with large data bases. The method uses the Monte Carlo method in conjunction with a hierarchical adaptive random partitioning scheme with fuzzy subdomains (MC-HARP). MC-HARP provides an environment for simultaneously building and training a mapping neural network. The method is self-organizing and can operate with minimal external adjustment. It can interactively accept knowledge and provide guidance for efficiently improving the model and the data base. The MC-HARP environment enjoys a large-scale granularity produced by the Monte Carlo parallelism and the geometric parallelism achieved by partitioning the input space.

We study the performance of the MC-HARP method by applying it to an experimental data base on pavement performance under a variety of environmental and traffic conditions. Numerical simulations are used throughout the report to demonstrate that the method is able to deal with high-dimensional, noisy, non-homogeneous data.

The MC-HARP method leads to a novel model selection criterion and an original framework for classifying data-fitting problems, and can be used to answer fundamental questions in data-based mathematical modeling. These questions include: What is the confidence level in the constructed model and the data base? What is the optimal functional structure of the model for noisy data? How appropriate is a particular parametric model for the given data? The MC-HARP method established an environment for unifying existing mathematical modeling techniques in statistics, approximation theory, information theory, system identification, and neural networks.

Mathematical modeling, multivariate mapping approximation, Monte Carlo method, neural networks, fuzzy subsets, divide-and-conquer method, model selection, empirical modeling, model validation, data fitting
Acknowledgements

The research reported herein was supported by the U. S. Army Research Office, as part of a program at the University of Illinois Advanced Construction Technology Center (grant numbers DAAL 03-87-K-006, DAAL 03-86-G-0186, DAAL 03-86-K-0188), and the National Science Foundation (grant number CES 86-58019). Joseph P. Murtha is the director of the Advanced Construction Technology Center and S.-C. Liu is the cognizant program officer at the National Science Foundation. The support of these agencies is gratefully acknowledged. The results, opinions, and conclusions expressed in this report are solely those of the authors and do not necessarily represent those of the sponsors.
# TABLE OF CONTENTS

## CHAPTER ONE: Introduction .............................................. 1
1.1 Engineering Applications of Data-based Mathematical Modeling ....... 3
1.2 Objectives and Scope .................................................. 9

## CHAPTER TWO: A Monte Carlo Strategy for Multivariate Mapping Approximation ........................................... 12
2.1 Local Approximation and Adaptive Partitioning ......................... 12
2.2 Subdomain Approximation ........................................... 15
  2.2.1 Parameter Estimators ........................................... 17
  2.2.2 Parameter Estimation Algorithms ........................................... 19
  2.2.3 Termination Criterion for the Subdomain Training ............... 21
  2.2.4 Variable Selection ........................................... 22
  2.2.5 Selected Subdomain Approximation Procedure .................. 23
2.3 Subdomain Partitioning ........................................... 25
  2.3.1 Selection of the Partitioning Function ........................ 28
  2.3.2 Selection of the Splitting Thresholds ........................... 29
  2.3.3 Number of Splits in a Subdomain .................................. 30
  2.3.4 Boundary Continuity ........................................... 32
2.4 HARP Algorithm for Multivariate, Mapping Approximation ........... 34
2.5 Numerical Simulations ............................................... 36
2.6 A Monte Carlo Strategy for Data-based Mapping Approximation ....... 39
2.7 Numerical Simulation for MC-HARP .................................. 41
2.8 Conclusions .......................................................... 42

## CHAPTER THREE: Numerical Simulation Studies of MC-HARP and Neural Network Representation of MC-HARP  ........................................... 45
3.1 Performance and Complexity Indices .................................. 45
3.2 Numerical Simulations ............................................... 47
  3.2.1 Effect of Sample Size ........................................... 47
  3.2.2 Effect of the Complexity of the Subdomain Approximation ....... 50
  3.2.3 Effect of Termination Tolerance .................................. 52
  3.2.4 Effect of Fuzziness and Schemes for Selecting Splitting Thresholds .................................................. 55
  3.2.5 Effect of the Dimensionality of the Input Domain ............... 61
3.3 MC-HARP and Neural Networks ........................................... 66
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>General properties of the beam models</td>
<td>69</td>
</tr>
<tr>
<td>3.1</td>
<td>Deviation and complexity measures for MC-HARP solutions for the XOR problem</td>
<td>59</td>
</tr>
<tr>
<td>5.1</td>
<td>Classification of data sets</td>
<td>131</td>
</tr>
<tr>
<td>5.2</td>
<td>(a) Features in the MC-HARP computable performance curves, (b) Classification of a data set with respect to adequacy</td>
<td>133</td>
</tr>
<tr>
<td>5.3</td>
<td>MC-HARP Performance analysis for constructed mappings</td>
<td>138</td>
</tr>
<tr>
<td>6.1</td>
<td>Designs for the factorial experiment of flexible pavements under single-axle traffic</td>
<td>146</td>
</tr>
<tr>
<td>6.2</td>
<td>Performance of constructed approximations for the training set</td>
<td>155</td>
</tr>
<tr>
<td>6.3</td>
<td>Test sections and their bounding training sections</td>
<td>158</td>
</tr>
</tbody>
</table>
### LIST OF FIGURES

2.1 Hierarchical adaptive partitioning of the input domain .............................................. 14
2.2 Squashing function .................................................................................................................. 25
2.3 Partitioning the range of $\psi(x)$ ............................................................................................ 26
2.4 Example of a partitioning function and its inverse images of intervals on the real line ......... 27
2.5 Gate function ........................................................................................................................... 27
2.6 Indexing of the splits and partitioned subdomains ................................................................ 29
2.7 Schemes for selecting splitting thresholds .............................................................................. 31
2.8 Splitting schemes ..................................................................................................................... 31
2.9 Smooth gate function for the jth interval in the range of $\psi(x; c)$ ....................................... 33
2.10 Fuzziness for a HARP partition ............................................................................................... 34
2.11 Solutions for the XOR problem .............................................................................................. 37
2.12 Evolution of a HARP solution for the two-dimensional surface fit problem ......................... 38
2.13 Learning curves of the HARP algorithm for the two-dimensional surface fit problem .......... 39
2.14 HARP partition for the two-dimensional surface fit problem ............................................... 39
2.15 Two HARP partitions developed by MC-HARP ................................................................. 41
2.16 Behavior of the MC-HARP method ....................................................................................... 43
3.1 Variation of performance and complexity indices with respect to the sample size for different numbers of data .............................................................................................................. 48
3.2 Variation of performance and complexity indices with respect to the number of data for different subdomain approximations ................................................................................ 51
3.3 Performance of MC-HARP with respect to the termination tolerance .................................... 53
3.4 Variation of the performance indices for MC-HARP approximations with fuzzy and nonfuzzy partitions with respect to the amount of data ................................................................................................................................. 56
3.5 Performance curves for MC-HARP approximations with fuzzy and nonfuzzy partitions ........ 56
3.6 Variation of the performance and complexity indices with respect to the amount of data for different schemes for selecting splitting thresholds .................................................. 57
3.7 Variation of deviation measure with respect to the sample size for the XOR problem ............ 58
3.8 MC-HARP solutions for the XOR problem ............................................................................. 60
3.9 Function $Tri(x)$ ....................................................................................................................... 62
3.10 Variation of performance and complexity indices for MC-HARP approximations with respect to the number of data for different dimensions of the input domain .......................... 63
3.11 Variation of performance and complexity indices for a HARP approximation with respect to the number of data for different dimensions of the input domain ........................................... 64
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.12</td>
<td>Neural network representations of $L^p$ functions</td>
<td>67</td>
</tr>
<tr>
<td>3.13</td>
<td>Global configuration of a HARP neural network</td>
<td>69</td>
</tr>
<tr>
<td>3.14</td>
<td>A HARP neural network</td>
<td>70</td>
</tr>
<tr>
<td>3.15</td>
<td>A HARP solution for the XOR problem</td>
<td>71</td>
</tr>
<tr>
<td>3.16</td>
<td>A HARP neural network with only sigma units</td>
<td>72</td>
</tr>
<tr>
<td>3.17</td>
<td>Intermediate HARP neural network $F_1$</td>
<td>73</td>
</tr>
<tr>
<td>3.18</td>
<td>Intermediate HARP neural network $F_2$</td>
<td>74</td>
</tr>
<tr>
<td>3.19</td>
<td>MC-HARP neural network</td>
<td>76</td>
</tr>
<tr>
<td>4.1</td>
<td>Actual function $G_{r_1}$</td>
<td>85</td>
</tr>
<tr>
<td>4.2</td>
<td>Performance curves for $G_{r_1}$</td>
<td>88</td>
</tr>
<tr>
<td>4.3</td>
<td>Performance curves for $G_{r_1}$</td>
<td>89</td>
</tr>
<tr>
<td>4.4</td>
<td>Performance curves for $G_{r_2}$</td>
<td>90</td>
</tr>
<tr>
<td>4.5</td>
<td>Performance curves for $G_{r_2}$</td>
<td>91</td>
</tr>
<tr>
<td>4.6</td>
<td>Variation of the approximation risk at $\varepsilon = \varepsilon_{\text{max}}$</td>
<td>93</td>
</tr>
<tr>
<td>4.7</td>
<td>Variation of the approximation error at $\varepsilon = \varepsilon_{\text{max}}$</td>
<td>94</td>
</tr>
<tr>
<td>4.8</td>
<td>Variation of the approximation risk at $\varepsilon = 0$</td>
<td>96</td>
</tr>
<tr>
<td>4.9</td>
<td>Variation of the standard deviation of approximation at $\varepsilon = 0$</td>
<td>96</td>
</tr>
<tr>
<td>4.10</td>
<td>Variation of the approximation risk with respect to the tolerance value for different numbers of data and amplitudes of noise</td>
<td>98</td>
</tr>
<tr>
<td>4.11</td>
<td>Variation of the number of subdomains for an MC-HARP approximation</td>
<td>99</td>
</tr>
<tr>
<td>4.12</td>
<td>Variation of the number of subdomains for the optimal approximation</td>
<td>100</td>
</tr>
<tr>
<td>4.13</td>
<td>Comparing the rate of increasing the number of subdomains for the optimal approximation</td>
<td>100</td>
</tr>
<tr>
<td>4.14</td>
<td>Variation of the optimal tolerance value</td>
<td>102</td>
</tr>
<tr>
<td>4.15</td>
<td>Behavior of the approximation risk with respect to the amplitude of noise and tolerance value for large number of data</td>
<td>103</td>
</tr>
<tr>
<td>5.1</td>
<td>Evolution of performance curves and quality-quantity map for the function $G_{r_1}$</td>
<td>109</td>
</tr>
<tr>
<td>5.2</td>
<td>Evolution of performance curves and quality-quantity map for the function $G_{r_2}$</td>
<td>110</td>
</tr>
<tr>
<td>5.3</td>
<td>Performance estimation techniques</td>
<td>113</td>
</tr>
<tr>
<td>5.4</td>
<td>Variations of the approximation risk and the approximation deviation for the optimal tolerance value</td>
<td>118</td>
</tr>
<tr>
<td>5.5</td>
<td>Quality-quantity map for a data set</td>
<td>130</td>
</tr>
<tr>
<td>5.6</td>
<td>Decision tree for classifying a data set</td>
<td>132</td>
</tr>
<tr>
<td>5.7</td>
<td>Performance curves</td>
<td>137</td>
</tr>
<tr>
<td>5.8</td>
<td>Complexity curves</td>
<td>139</td>
</tr>
<tr>
<td>5.9</td>
<td>Performance of constructed MC-HARP approximations</td>
<td>140</td>
</tr>
</tbody>
</table>
5.10 Performance of the best $x_1^2x_2x_3x_4$ mapping for the test set ............. 141
6.1 Pavement structure ........................................................................... 143
6.2 Variation of the Smooth Data .............................................................. 151
6.3 Smooth serviceability trends and trends predicted by AASHO formula .... 152
6.4 Performance of the AASHO formula for the training set ....................... 154
6.5 Performance of the MC-HARP approximation with linear subdomain approximations for the training set $\varepsilon=0.1$ .......................................................... 160
6.6 Performance of the MC-HARP approximation with linear subdomain approximations for the training set $\varepsilon=0.1$ .......................................................... 162
6.7 Performance of the MC-HARP approximation with linear subdomain approximations for the test set $\varepsilon=0.1$ .......................................................... 163
6.8 Performance of the filtered MC-HARP approximation with linear subdomain approximations for the test set $\varepsilon=0.1$ .......................................................... 165
6.9 Performance curves for the MC-HARP approximation with linear subdomain approximations .......................................................... 166
6.10 Performance of the MC-HARP approximation with linear subdomain approximations for the training set $\varepsilon=1.75$ .......................................................... 166
6.11 Performance of the MC-HARP approximation with linear subdomain approximations for the training set $\varepsilon=1.75$ .......................................................... 168
6.12 Performance of the MC-HARP approximation with linear subdomain approximations for the test set $\varepsilon=1.75$ .......................................................... 171
6.13 Performance comparison of the MC-HARP approximations with linear subdomain approximations .......................................................... 167
6.14 Performance curves for the MC-HARP approximation with the AASHO subdomain approximation .......................................................... 176
6.15 Complexity curves for MC-HARP approximations .................................. 177
6.16 Performance of the MC-HARP approximation with the AASHO subdomain approximation for the training set $\varepsilon=1.75$ .......................................................... 177
6.17 Performance of the MC-HARP approximation with the AASHO subdomain approximation for the training set $\varepsilon=1.75$ .......................................................... 179
6.18 Performance of the MC-HARP approximation with the AASHO subdomain approximation for the test set $\varepsilon=1.75$ .......................................................... 182
6.20 Distribution of parameters in the AASHO formula .................................. 186
A.1 A neural network .............................................................................. 195
A.2 Schematic processing unit .................................................................. 196
A.3 A feedforward neural network .............................................................. 197
A.4 Different kinds of output functions ...................................................... 199
A.5 A hidden unit in a backpropagation network ........................................ 204
CHAPTER ONE

Introduction

"Rationally, let it be said in a whisper, experience is certainly worth more than theory."

Amerigo Vespucci

"So far as the laws of mathematics refer to reality, they are not certain. And so far as they are certain, they do not refer to reality."

Albert Einstein

Recent progress in computer and other high technology industries has made the gathering of information and data easier. Global change studies, astronomy, human genome mapping, social and economic studies, and engineering design are a few examples of research areas that generate or require access to extraordinarily large amounts of data. Data-based models have far-reaching potential as building blocks in tomorrow’s computational world. There are many applications for which data-based information systems appear to offer a more appropriate approach to computing than some of the more traditional approaches. Data-based systems show great promise for solving problems that require pattern recognition, pattern mapping, filtering noise, pattern completion, and learning or adaptation during use. Since the beginning of the 1980s, a large amount of research in parallel, distributed, data-based models has focused on practical applications in addition to their use as models of human thought processes. These parallel, fault tolerant, and computationally efficient models are used to perform difficult tasks, especially those for which the classical rule-based approaches, theoretical modeling, and system identification fail to produce acceptable results.

Very few systems can be adequately modeled using theory alone. There are always parameters in a theoretical model whose values must be assumed or empirically determined. Testing and experiments often provides valuable information that a theory cannot provide. Often existing theoretical models are not able to reconcile the acquired data to the desired level of accuracy. The gap between theory and experiment has pushed researchers to look for methods to improve their mathematical models. Data-based mathematical modeling is a powerful alternative to theory and, in many cases, the only approach to knowledge representation.

For example, consider the problem of modeling pavement deterioration for the purpose of making decisions regarding the maintenance, repair, or replacement of a network of roads. A pavement is a layered structure designed to distribute traffic loads to the subgrade soil. A pavement deteriorates due to physical and chemical processes in the materials and are driven by environmental effects and traffic. The structural behavior of a pavement is a function of the geometric and material properties of its layers and the subgrade. Traffic is a variable load that is affected by the traffic volume and its history, as well as vehicle characteristics such
as weight, speed, tire pressure, axle spacing, and suspension. Environmental conditions include temperature, humidity, precipitation, and the presence of aggressive chemicals and vary in an uncontrolled fashion and inflict damage to the pavement. The amount and variability of factors affecting a pavement make its behavior extremely complex.

Because of uncertainties in and complexity of modeling environmental effects, traffic, and the mechanical behavior of pavements, development of a theory of pavement behavior would be difficult, if not impossible. However an engineer must still make decisions concerning the design, maintenance, and repair of these systems, and hence he needs a model of pavement deterioration. One approach to this problem would be to monitor the performance of pavement sections under controlled conditions, and thereby develop a performance database. The database could then be used to develop an empirical model of pavement performance that could be used to predict deterioration in other circumstances. Actually, such an experiment for modeling a pavement system was conducted by the American Association of State Highway Officials (AASHO) during the period of 1958 through 1960. The AASHO Road Test has been the basis for pavement design practices during the last thirty years. We will take a closer look at the AASHO Road Test later.

The upshot of this particular example is to point out that there are many real engineering problems for which the development of theories of behavior are lagging behind the need for answers. For many of these problems, data are available. Presumably these data contain knowledge of the behavior of the system. The only question is how to access this information. The lack of a theoretical model and existence of uncertainties suggest using data-based mathematical modeling for such problems. In the next section, we describe a number of engineering applications for which this approach is applicable.

The need for a portable, practical data processing tool for these complex and diverse engineering problems is evident. Current data analysis technology falls short of offering such a numerical tool. Neural networks are the most recent technology to show promise in filling the needs of the data-based modeler.

Neural networks use a parallel, distributed processing structure comprising a set of simple, interconnected processors. This processing system can organize itself to build a mathematical model to represent the knowledge contained in the data. During the training process, the parameters of a neural network are adjusted to represent the mapping implicit in the data. The trained neural network generalizes its understanding of the data used to train it, by producing results for new cases. Besides this self-organization and learning feature, neural networks possess a parallel structure that has a powerful potential for creative hardware implementations of massively parallel processors. Neural network area of research promises of a data processor with self-organization capability, adaptivity, generalization, parallel distributed computation, noise and fault tolerance, and speed. A neural network developed with current technology is hampered by a number of obstacles that impairs its usefulness. These obstacles include slowness in training, the difficulty of selecting a suitable network configuration for a particular task, and the stability and plasticity of the system with respect to the introduction of new data. Furthermore current neural networks cannot answer certain fundamental questions in data-based mathematical modeling. These questions include: Is the amount of training data adequate? If not, which regions of the input space need more data? What confidence should one have in the constructed model, and how might one quantify that confidence?
Despite these shortcomings, neural networks have made two important contributions to engineering. First, they provide a bridge between lines of research and application that have been separated for years. Researchers from different disciplines like statistics, physics, information processing, computer science, engineering, and biology have started to communicate and transfer their technologies. A contribution to research in neural networks often influences researchers in vastly different fields. Second, neural networks have revealed some important characteristics of data processing system that is needed for diverse applications in diverse areas that traditional approaches to data processing have been unable to provide. In the following paragraphs we attempt to enumerate the characteristics that are needed for a general data-processing tool. Once described we shall proceed to develop just such a tool.

A desirable data-processing tool would provide a complete environment for building data-based mathematical models. The environment would have a solid mathematical foundation and would be implemented in a computer program. Such an environment could assist researchers in many disciplines to better utilize their data by providing a powerful mathematical tool for building data-based mathematical models and information processing systems. It would allow them to spend more time gathering information and less time on the process of building mathematical models. The environment should allow a researcher to quickly determine whether data-based mathematical modeling is promising in his or her particular case or not.

One of the main challenges in developing such a modeling environment is to endow it with a robust and computationally efficient training process for large and complex data sets, while requiring minimal user interaction. In order to have a robust training process, the training environment should build parsimonious functional configurations with good generalization, it should reduce the probability of constructing spurious configuration, it should have highly plastic and stable learning characteristics with respect to the addition of new data points, and it should be consistent in reducing the bias between the constructed mathematical model and the actual model as the size of the training set increases.

With regard to computational efficiency, the environment should not only build fast data-based mathematical models that take advantage of the massively parallel architectures but it should also have a highly parallel training strategy that would enable it to exhibit good speed-up and scalability on multiprocessor computers and for on-line computations. The environment should be able to deal with highly complex databases, including such considerations as noise, high dimensionality, a mixture of data types, nonstandard data structures, and nonhomogeneity (that is, different relationships hold between variables in different parts of the measurement space). The latter source of complexity arguably presents the greatest challenge.

Constructing a functional representation for complex tasks based on scattered measurements is an important research topic in fields like approximation theory, neurocomputing, system identification, information theory, and statistics. The training environment should be comparable with, compatible with, or convertible to existing algorithms if other approaches exist. The new environment should do at least as well as parochial tools when applied to "solved" problems, but should provide a vehicle for travelling uncharted territory.

Problems in mathematical modeling, forecasting, design, pattern recognition and classification, diagnosis, signal processing, image processing, neural computers, control, robotics, and numerical approximation

3
are among the topics that could benefit from the proposed data-processing tool. In the following section we describe how certain engineering applications might benefit from the development of such a tool.

1.1 Engineering Applications of Data-based Mathematical Modeling

There are many engineering applications that use or could use data-based mathematical models. Engineers interpret data measured by sensors to detect defects in, assess the condition of, simulate and predict performance of, and/or control an engineering system. Engineers use measurements from experiments to build empirical models for analyzing and designing engineering systems. Engineers use data processing tools to interpret results generated by their mathematical models. To illustrate the applicability of data-based models, we describe some of main applications of such tools with more emphasis upon engineering applications.

Application to Mathematical Modeling. Model building is a fundamental task in the natural sciences and engineering because of the importance of experiments and measurements in these fields. Observing the behavior of a system and measuring its input and output are essential to building a mathematical model of the system. In general, model building is a problem of mapping a set of observations to a set of candidate models. This process is usually referred to as system identification. The identification process tries to construct a model for a system or improve the existing model of a system based on available observations, and any prior knowledge about the system. The identified model provides physical insight about the system and can be used to simulate or predict the response of the system.

To build a mathematical model we begin with generally accepted physical laws (Newton’s laws, Maxwell’s laws, Kirchhoff’s laws, mass balance, energy/heat balance) for the system or process being studied. From these laws, a number of relations among the inputs and outputs of the system follow and establish the structure of the mathematical model. These relations often take the form of algebraic equations, ordinary or partial differential equations, or integral equations. If all external and internal conditions of the system are quantitatively known and if the physical knowledge about the system is complete then, at least in principle, the numerical values of all parameters in those relations can be determined using measurements. The constructed parametric mathematical model is then verified by observing its performance for new cases.

In this model building procedure, the model structure is selected on the basis of prior knowledge about the system and its purpose. Only the functional form of the model needs to be selected in advance and the parameters of the model are estimated to minimize the discrepancy between the predictions of the model and the observations. A pure theory has no parameters that require fitting to observation. Such theories are hard to come by. Most natural systems have aspects for which the theory is well developed and aspects for which the theory is poorly developed or nonexistent.

A parametric model makes use of the available prior knowledge and uses the data to make the fine adjustments. The governing equations, based upon those things that we know well, provide the functional structure for our mathematical model. The aspects of the model that we do not know are parameterized and left to be estimated from the data. Whenever prior knowledge for selecting the model structure and the data for estimating parameters are available, a parametric mathematical modeling is the best approach. Unfortunately, for real systems, finding an appropriate mathematical model based on this approach may be difficult because
information is limited by incomplete or uncertain knowledge of the environment or the physical aspects of the system. Also, the consideration of many factors can make the model very complex. A solution for this problem is to estimate the functional form of the model completely from the data. We refer to this approach as the data-based mathematical modeling or empirical modeling. Data-based modeling is usually referred to as nonparametric modeling as opposed to the parametric modeling.

The only information that a purely data-based mathematical model needs is data from the actual system. The model does not explicitly require knowledge of the physical aspects of the process it is trying to represent. In a parametric mathematical model just enough data is needed so that the controlling parameters of the model can be identified. On the other hand, empirical modeling requires a comprehensive data set that covers the input-output domain of a system with a reasonable resolution. A purely data-based mathematical model has more flexibility than a parametric or purely theoretical model at the price of requiring much more data. Mathematical models represent a wide spectrum of approaches to engineering problems. Purely theoretical modeling is at one end of the spectrum and purely empirical modeling is at the other end. The utility of a model depends on the amount of data and prior knowledge that are available.

Applications of data-based mathematical models in engineering include: (1) nondestructive testing methods for complex structural systems like buildings, pavements, bridges, piles, and aircraft, (2) system modeling for subsequent use in simulation, design, evaluation, and control studies of deformable mechanical systems, (3) condition monitoring of machines to enhance the efficiency of their maintenance and operation, (4) rehabilitation of structural systems, (5) constitutive modeling for new materials, and (6) modeling of environmental loads like earthquake, wind, and water waves. The following two applications in pavement design and evaluation are illustrative.

1. As mentioned earlier, the amount and variability of factors affecting a pavement make its behavior rather complex. The performance of a pavement is influenced by its structural properties, traffic, and environmental effects. A database can be developed by monitoring the performance of a broad range of existing pavement sections in-service. The gathered database is large, complex, and heterogeneous, including traffic and environmental data, pavement surface images, pavement profiles, and structural measurements. One could use the database in pavement design. Empirical models that are functions of traffic, climate, and structural factors can be developed in order to predict the deterioration of pavements. In Chapter Six we show the application of our proposed method for building data-based mathematical models to the problem of modeling pavement performance.

2. Nondestructive testing with a falling-weight deflectometer, is a fast and efficient means for collecting displacement data from which the material properties of pavement layers can be determined. Efficient and economical methods for determining the structural properties of existing pavements are of significant benefit in developing a good pavement management plan and can be a useful tool in developing a specific pavement rehabilitation program. Existing nondestructive test databases can be used to build a mathematical model for estimating structural properties of pavements. The constructed mapping would take the measured surface displacements and thickness of layers as its input values and would compute the material parameters of pavement layers as its output values.
**Application to Information Processing and Knowledge Representation.** An information processing produces outputs corresponding to information extracted from the input data. The functional specification of the transformation between inputs and outputs of an information processing system can be represented by mathematical formulas, a set of equations and constraints, rules, and a computer program. A data-based mathematical model is an information processing system.

The need to represent and generalize from information that is embodied in distributed patterns rather than a discrete set of them is a good motivation for using a data-based mathematical model. These models have the means for interpreting the rules implicit in patterns of information that traditional rule-based and parametric modeling approaches fails to express. There are many engineering applications where there is no way besides data-based modeling to approach this task. For example, the task of nondestructive testing of structural members using sonic or radiating waves. In these applications the wave propagation phenomenon is too complex to be modeled, and yet the monitored signals seem to contain useful information that can be processed.

Data-based components of an information system can serve as performance enhancement replacements for existing system elements. For example, data-based material models could enhance or replace the material constitutive library of a finite element program. Furthermore data-based techniques have the ability to learn rules more flexibly than rule-based knowledge-processing techniques. More sophisticated information systems may be built from a combination of rule-based and data-based systems. For example, in the case of a finite element code we implement the rules of computation by programming and use a data-based mathematical model to import knowledge from a material database into the program.

Implicit knowledge is difficult to quantize, formalize, or sometimes even express verbally. Furthermore translation of implicit knowledge into explicit rules may lead to loss and distortion of information content. Therefore domains with implicit knowledge are difficult to express in terms of accurate rules. Data-based information systems can acquire knowledge without extracting rules from a human expert provided that the number of training patterns is sufficient. Thus data-based techniques would be able to ease the knowledge acquisition bottleneck that is hampering the development of conventional expert systems. In comparison to conventional symbolic methods, data-based models have advantages in robustness, computational speed, and dealing with uncertain and conflicting evidence for high level artificial intelligence problems like graph matching and constraint propagation. A data-based modeling technique can complement conventional information systems. Furthermore a hybrid system using rule-based and data-based approaches is more feasible for practical applications of information systems than a system using one of these techniques alone.

**Application to Pattern Recognition and Classification.** One of the most important results of learning from a set of exemplary cases is the ability to recognize patterns and classify them. A pattern is a feature vector describing an object, event, or phenomenon. The classification may involve spatial patterns like images and weather maps or temporal patterns like signals and seismograms. A set of patterns whose classes are explicitly known are used to train a data-based classifier. The classifier takes the pattern associated with an object as its input and assigns the object to one of the prescribed classes. For example in a diagnostic task, a feature pattern corresponds to symptoms and classes are assigned to particular diagnoses. A data-based classifier is
a mapping from the space of patterns to the space of classes. The following applications of data-based classification are exemplary.

1. Currently, a great deal of human effort is required to recognize and measure defects in pavement surfaces in order to assess the condition of the pavement. The current practice for pavement condition assessment is too time consuming and costly for monitoring pavements of the highway system. Surface images are used to automatically recognize pavement features from a moving survey vehicle and classify its condition. A surface image contains feature vectors associated with different classes of surface defects. A set of surface images with known condition ratings could be used to train a data-based classifier. The classifier could learn the classification task from a database of surface images.

2. The concern for studying man's impact on the landscape has led to a large investment in remote sensing techniques such as satellite imagery. Numerous satellites provide an ever increasing flow of multi-spectral image data. The existing satellite image classifiers require a great deal of human interaction from highly trained professionals. A parallel robust image classifier could process satellite imagery for assessing the land coverage condition, with a speed compatible with the rate of data transmission from satellites.

3. Data-based classifiers could be used for classification of pavement sections with respect to their rideability. It is far too complicated, time consuming, and expensive to rely on subjective rating for assessment pavement condition. One solution is to correlate objective physical measurements of pavement characteristics with the subjective public's perception of rideability. A data-based classifier could predict subjective panel ratings of pavement rideability from profile measurements of pavement roughness. Measured profiles of pavement sections with known rideability ratings could be used to train the classifier. Here a data-based mathematical model behaves as a signal classifier mimicking human response to road roughness, replacing a panel of raters.

**Application to Fault Detection and Diagnosis.** A system is operating normally when its controlled variables are in the neighborhood of their desired values. Damage and deterioration modify the response of a system under normal working conditions. A fault occurs when a certain level of deterioration takes place in performance of the system because of permanent physical changes or random fluctuations of values for controlled parameters. A fault diagnosis system should detect a fault as soon as possible, localize the fault, and identify the physical causes of the fault. The changes in the response might be associated to specific damage states. Building response-based inspection systems is a difficult task. However, it might be possible to learn the associations between faults and changes from examples. A data-based mapping approximation could be used for pattern classification problems including fault detection and diagnosis. A data-based fault diagnosis technique could be developed by constructing a mapping from the space of sensor data (fault symptoms) to a class of associated faults.

Many billions of dollars are lost every year due to inadequate maintenance of structures. The cost of interrupted services after a natural disaster could be astronomical. New methods for the inspection of structural condition need to be developed to help manage these maintenance problems. Data-based inspection systems have been used for structural monitoring of load carrying systems such as aircraft, space structures, buildings, bridges, offshore platforms, underground structures, and mechanical systems. In these structures, attempts
have been made to assess structural damage from changes in the response of the structure to ambient and impact loads or from signals recorded using ultrasonic, microwave, and radiating waves. Engineers have been attracted to response-based damage detection methods because of the extreme difficulty and expense of inspecting complex structures especially those which are hard to access like underwater, underground, or space structures.

The response of an structural system like a pavement structure to external excitations like a falling weight could be used to assess its condition and detect particular defects. A data-based structural defect detector for pavements would take the response of a pavement to an external excitation as its input and would classify pavement defects, if there are any, as its output. Another example of applications of data-based diagnosis systems is in the development of wayside methods of fault detection in railway engineering. The basic concept is to build a classifier to identify the existence of faults in bearings and wheels in moving trains from the acoustic signals measured by wayside microphones. A number of recorded signals are used to train the classifier. The trained classifier would then take the acoustic signal associated with a wheel or bearing as the defect symptom and diagnose the condition of the bearing or wheel.

Application to Numerical Approximation. The Finite Element (FE) method is a numerical analysis technique for obtaining approximate solutions to many of the partial differential equations of mathematical physics. The efficiency of finite element (FE) codes could be improved by fitting them with parallel data processors. Here a data-based mapping approximation could be used in three parts of a FE code: (1) post-processing step, (2) material modelling, and (3) error estimation for adaptive solution refinement.

1. A FE program generates a data set containing the solution values at a set of discrete points. Post-processing techniques used to compute the values of a FE solution and its derivatives at other points in the problem region are not very accurate and researchers need to know these values throughout the domain. A parallel data-based function approximator could accurately and quickly interpolate a FE solution and its derivatives.

2. The material modeling subprogram of a FE code contains a library of mathematical models representing the constitutive relations among secondary variables like strains and stresses. These material models are highly nonlinear and compute their outputs through sequential, iterative calculations that occupy a significant portion of the computation time. These sequential, iterative material models can be approximated and replaced with vectorized, parallel, noniterative models using data-based mapping approximation methods to speed up nonlinear FE analyses.

3. Data-based mapping approximators can provide a reliable error estimator for a FE solution and its derivatives based on their values at discrete points in the problem region. The error estimation could then guide the FE code to adaptively refine its approximate solution in regions with large errors.

Increasing the accuracy of post-processing, adding robust adaptivity, and speeding up material modeling can significantly improve the efficiency of FE codes on parallel computers. These improvements are also useful for other numerical approximation methods like the Boundary Element and Finite Difference methods and solution strategies like multigrid methods. Furthermore, data-based approximations can fit the output of a complex computer code, so as to both approximate and understand the code. Approximation of such a code is often necessary since it is slow or is too expensive to be run for all intended purposes.
Application to Control. Control is the set of actions taken to make a system behave in a satisfactory manner. The problem of control in general can be considered as a mapping from the space of measured system response to the space of corrective actions. The controller manipulates inputs to the system, based on the state of the system, so that the outputs achieve certain specified objectives. A system control problem is a very challenging task especially when the system and its environment are complex, varying, uncertain, or difficult to simulate and predict.

Controlling a system involves two fundamental processes: modeling of the system based on the information provided by sensors and determining the control inputs. A data-based mathematical model with capabilities like self-organization, noise and fault tolerance, adaptivity, and parallel structure has potential applications at all stages of a control system. One could use a data-based approach to model the dynamics of the system, the system inverse, or the controller as an alternative or replacement for parametric mathematical modeling. Such an approach would be especially attractive when there are significant uncertainties and variability in the system model and the environment. The parallel structure of the approximate mapping has the advantages of speed for real-time operations and fault tolerance to accommodate defective hardware. Furthermore a data-based information system as a knowledge representer can link human expertise with the control system.

A data-based mathematical model that can adjust its functional form via training is attractive for adaptive process control. Its self-modeling and organization capabilities allow a process to be remodeled quickly or to adjust the controller for varying process and environmental conditions or degraded equipment. A data-based approximate mapping can mimic an existing controller and replace it if the controller currently in use is expensive or unreliable. Furthermore a data-based mathematical model could be used in parallel with an existing controller to enhance its operation when the control action has deteriorated because of age or damage to the plant. After it has replaced the current controller, the data-based model could be adjusted through training to take into account changes in the plant and the environment.

Control of complex systems represents a problem of fundamental importance in a wide variety of engineering applications. Examples of control systems are active controllers for deformable bodies and robots, the autopilots in airplanes, controllers for multiple autonomous undersea vehicles, the pointing mechanisms of space telecommunication antennas, flow controllers for raw materials in an industrial plant to yield a desired product, speed regulators of machines, controllers for emissions control and suspension systems in automobiles, controllers for temperature and humidity regulators in buildings.

1.2 Objectives and Scope

We have shown that a data-based mathematical model with self-organization capability, noise and fault tolerance, adaptivity, generalization, and parallel structure has a large number and vast range of potential applications in engineering. A desirable data-processing environment should have a fast and parallelizable training process. It should be able to process heterogeneous data and operate while requiring minimal external adjustment. It should interactively accept prior knowledge and guide the experimenter in efficiently improving the database and the constructed mathematical model. Current data analysis technology falls short of of-
fering such a numerical tool. In this work we develop such an ideal training environment and data-based mathematical model. The data-based mathematical model and training environment proposed herein can support the diverse need of scientific and engineering applications.

This manuscript consists of seven chapters and one appendix. In Chapter Two, we develop a novel training environment for building data-based mathematical models called MC-HARP. A Monte Carlo (MC) strategy combined with the concept of Hierarchical Adaptive Random Partitioning (HARP) and fuzzy subdomains determines the functional structure and parameters of a mathematical model simultaneously. We describe the main processes of the MC-HARP method namely, the subdomain approximation, subdomain partitioning, fuzzy smoothing, and Monte Carlo sampling. We illustrate the MC-HARP methodology through numerical simulations.

In Chapter Three, we study the performance of an MC-HARP approximation through numerical simulations. We establish procedures and rules for selecting the main components of the MC-HARP method namely, the subdomain approximation, the number of partitions, the subdomain partitioning schemes, the continuity and boundedness modifications. The behavior of an MC-HARP approximation with respect to the dimension of the input domain is studied. We demonstrate that the rate of convergence for an MC-HARP approximation is independent of the dimensionality of data. Further, we show that the MC-HARP method can be used to simultaneously train and build mapping neural networks. The training process of MC-HARP is shown to be compatible with the training process of a mapping neural network. We show that an MC-HARP approximation can be modeled as a modular feedforward neural network.

In Chapter Four, we use numerical simulations to study the performance of the MC-HARP method for noisy data. Performance indices are defined to investigate the complexity-dependent accuracy of an MC-HARP approximation for different noise amplitudes and amounts of data. We extract general trends in the performance of constructed MC-HARP approximations to establish a framework for performance analysis of MC-HARP. Furthermore, we demonstrate that the tolerance value for the termination criterion for subdomain training process is the only complexity-controlling parameter for an MC-HARP approximation. The existence of an optimal tolerance value corresponding to an approximation with the optimal complexity and lowest approximation error is shown and we provide a concept for selecting the optimal tolerance value.

In Chapter Five, we propose a new model selection criterion. The proposed MC-HARP model selection criterion is based on the minimization in the limit of the deviation measure, computed by MC-HARP, with respect to the amount of data over the entire input domain. We express the MC-HARP philosophy for performance estimation of data-based approximate mappings and illustrate its advantages to sampling-based techniques. We define quantitative measures for approximation confidence and accuracy of an MC-HARP approximation and also the adequacy of data. We establish a novel MC-HARP framework for classifying data-fitting problems with respect to the quality-quantity conditioning of their data sets. We show how the proposed framework is able to detect an ill-conditioned data-fitting problem and to warn the experimenter that the performance of the constructed mathematical model may be unreliable. We use a real data-set to illustrate the application of the proposed MC-HARP model selection technique and framework for classifying nonparametric, data-fitting problems.
In Chapter Six, we show that a data-based mathematical model can be built to model performance of a pavement. We use the MC-HARP method with the MC-HARP model selection technique and framework for classifying data sets to build an empirical model for pavement performance using the data of the AASHO Road Test. We demonstrate the superiority of the MC-HARP model over the AASHO model currently used for pavement design. We show that prior knowledge about the physics of the problem and the actual model can be used to improve generalization and reliability of a data-based mathematical model. Furthermore we illustrate how MC-HARP can be used to verify an existing parametric mathematical model.

Chapter Seven is a summary.

In Appendix A, we review the concept of neural networks, elucidating their structure, and describing how they process information and self-organize.
CHAPTER TWO
A Monte Carlo Strategy for Multivariate Mapping Approximation

"Beauty of style and harmony and grace and good rhythm depend on simplicity."

Plato

"Cottages may be built without modelles, not pallaces."

Sir W. Cornwallis

2.1 Local Approximation and Adaptive Partitioning

Our main goal in this chapter is to propose a method for finding a solution for the following function approximation problem:

Let \( \mathcal{D} \) be a closed bounded subset of \( \mathbb{R}^n \) and let \( G \) be a real-valued function defined on \( \mathcal{D} \) with values \( y_i = G(x_i) \) given at a set of \( N \) points \( \{x_1, x_2, ..., x_N\} \), with each \( x_i \in \mathcal{D} \).

Find a function \( F: \mathcal{D} \subset \mathbb{R}^n \rightarrow \mathbb{R} \) that reasonably approximates \( G \) and belongs to the collection of functions \( \mathcal{L}_\mathcal{D} \).

The set \( \mathcal{L}_\mathcal{D} \) is the collection of multivariate functions \( f: \mathcal{D} \subset \mathbb{R}^n \rightarrow \mathbb{R} \) which can be written as the scalar summation, scalar multiplication, and composition of a finite collection of univariate functions. The collection \( \mathcal{L}_\mathcal{D} \) contains a vast class of functions including polynomials, splines, trigonometric functions and their byproducts generated by summation, multiplication, and composition of these functions.

Based on the Stone-Weierstrass theorem (Royden 1988), every continuous function on \( \mathcal{D} \) can be uniformly approximated on \( \mathcal{D} \) by a polynomial. Since \( \mathcal{L}_\mathcal{D} \) contains the space of polynomials, \( \mathcal{L}_\mathcal{D} \) is dense in the space of continuous functions. Also, Kolmogrov's theorem (1957) proves that any continuous multivariate function on \( \mathcal{D} \) can be approximated as a summation and composition of a finite collection of continuous univariate functions. The Kolmogrov approximation belongs to the collection \( \mathcal{L}_\mathcal{D} \). Hence for each continuous function on \( \mathcal{D} \) there exists a function (Kolmogrov approximation) in the collection \( \mathcal{L}_\mathcal{D} \) that approximates the given function. In other words, the Kolmogrov's theorem proves that \( \mathcal{L}_\mathcal{D} \) is dense in the space of continuous functions.

The collection \( \mathcal{L}_\mathcal{D} \) contains several spaces of functions including polynomials, splines, radial basis functions, and trigonometric functions which can approximate any continuous function on \( \mathcal{D} \). In contrast with the mainstream in function approximation theory, which tries to narrow down the space of approximators, here we expand our space of approximators \( \mathcal{L}_\mathcal{D} \) to contain a large class of functions. The expansion is a natural result of the weak restrictions that apply to the functions in \( \mathcal{L}_\mathcal{D} \), restrictions that are inspired by issues...
of implementation. We want the constructed approximator implemented on a computational machine that is capable of doing a finite number of simple operations of addition, multiplication, and composition of univariate functions. Our computational machine is a neural network and we will show in the next chapters how one can represent a function in \( \mathcal{L}_s \) by a neural network. Actually we will explain that any function in \( \mathcal{L}_s \) can be written as the scalar summation and composition of a finite collection of univariate functions and that scalar multiplication is a redundant operation for defining \( \mathcal{L}_s \). However for the sake of clarity, the definition of \( \mathcal{L}_s \) is set as above until we change it.

One can assume boundedness, continuity, and smoothness constrains of the function \( G \) to assure the convergence and consistency of the approximation algorithm and to exclude the possibility of approximating a completely random process. From now on we will refer to \( \mathcal{G} = \{ x_i \}_{i=1}^N \) as the set of training data points, \( \mathcal{Y} = \{ y_i = G(x_i) \}_{i=1}^N \) as the set of target values, \( \mathcal{T} = \{ (x_i, y_i) \}_{i=1}^N \) as the training set, and \( \mathcal{D} \) as the input domain.

Also for the sake of clarity we will explain our proposed method for solving the function approximation problem with noise-free target values and we will discuss how to improve the proposed method to handle noisy data.

There are basically two approaches to handling a data fitting problem: local methods and global methods. Local methods construct the function \( F \) such that the value at any point depends only on the data at relatively nearby points. The value of the function in a global method is influenced by all of the data. The estimation of the parameters of \( F \) in a global method requires the solution of a fairly large system of equations, whereas local methods generally require the solution of a large number of small systems of equations. Local methods are more readily adapted to complicated functional behavior than are global methods.

The main character of our proposed method is local approximation and can be stated as

\[
F(x) = \theta_i(x) \quad \text{if} \quad x \in \mathcal{D}_i \quad i = 1, ..., s
\]  

where \( \mathcal{D}_i \) is the \( i \)th subdomain in the partition \( \mathcal{C} \), where

\[
\mathcal{C} = \{ \mathcal{D}_i : \mathcal{D} = \bigcup_{i=1}^s \mathcal{D}_i, \mathcal{D}_i \cap \mathcal{D}_j = \phi \quad \text{if} \quad i \neq j \}
\]  

There are \( s \) subdomains and \( \theta_i \) is the locally supported approximation function for the \( i \)th subdomain.

To make the approximation in Eqn. (2.1) a member of \( \mathcal{L}_s \), the characteristic function of the subdomain \( \mathcal{D}_i \) can be used to change the conditional statement in Eqn. (2.1) to a function multiplication as follows:

\[
F(x) = \sum_{i=1}^s \chi_{\mathcal{D}_i}(x) \theta_i(x)
\]  

where the characteristic function \( \chi \) is defined by

\[
\chi_{\mathcal{D}_i}(x) = \begin{cases} 
1 & \text{if} \quad x \in \mathcal{D}_i \\
0 & \text{otherwise}
\end{cases}
\]
The functional form $F$ of Eqn. (2.3) is a summation of a finite collection of products of multivariate functions $\{\chi_\mathcal{D}(\alpha)\}$ and $\{\theta_i\}$. Therefore $F$ belongs to $\mathcal{L}_\mathcal{D}$ if the functions $\{\chi_\mathcal{D}(\alpha)\}_{i=1}^s$ and $\{\theta_i\}_{i=1}^s$ belong to $\mathcal{L}_\mathcal{D}$. In the following sections, we will show how to build the subdomain approximation function $\theta$ and the characteristic function $\chi$ such that they belong to the collection $\mathcal{L}_\mathcal{D}$.

The local approximation is built by a hierarchical adaptive random partitioning (HARP) algorithm in the following manner. For a given subdomain, fit the approximation function $\theta$ to the data in that subdomain. If the fit is not acceptable then randomly partition the subdomain and fit $\theta$ to the offspring subdomains. Continue the process until the approximation error in each subdomain is adequately small.

This algorithm is shown schematically in Fig. 2.1. Intermediate partitions of the input domain are horizontal lines of boxes and are indexed with a superscript, as in $C^v$ for the $v$th intermediate partition. The final partition $C^{v_{\text{max}}}$ is designated simply as $C$. The number $v_{\text{max}}$ is referred to as the depth of the partitioning tree. A shaded block means that the fit by $\theta$ is acceptable. The shaded blocks are darkened for the first intermediate partition in which the fit is acceptable. The construction of the domain partition $C$ is hierarchical because each subdomain contains its offspring subdomains. The partitioning scheme has the flexibility to adapt to the complexity of the data so that more subdomains are generated where the data have complex behavior.

If the input domain $\mathcal{D}$ is partitioned such that each subdomain contains at least one training data point, then the HARP algorithm is guaranteed to find a local approximation as defined in Eqn. (2.2) for the approximation problem. The worst case is when $\theta$ is a constant function and the partitioning is continued until there is only one data point in each subdomain which can be fit exactly by the constant function. Therefore in general, the depth of partitioning tree $v_{\text{max}}$ and the number of subdomains $s$ in the final partition (final subdomains) are finite when the training set is finite.

There are two major issues to be addressed in the proposed algorithm. First, how does one choose the approximation function $\theta$ and fit it to the data and second, how does one randomly partition a subdomain and build its characteristic function $\chi$. The complicating issue is that the solution of these problems; i.e., $\theta$ and $\chi$, must conform to the properties of the collection $\mathcal{L}_\mathcal{D}$.

---

**Fig. 2.1 Hierarchical adaptive partitioning of the input do-**
2.2 Subdomain Approximation

The subdomain approximation function $\theta$ belongs to a collection $\Theta$ of parametric functions that are defined in terms of a finite number of unknown parameters $w$. In other words, to fit the data in each subdomain (local adaptation), we assume that the form of the approximation function $\theta$ is known except for a finite number of parameters computed by minimizing an index of the error between the computed output values $\theta(x; w)$ and the expected output values $y$. The experimenter may choose one possible family of functions when theory, past experience and/or other sources are available that provide detailed knowledge about the form of the actual mapping $G$. We will discuss in more detail how a priori knowledge of the physics of the problem can be used to restrict the structure of $\theta$ when we process the data of the AASHO Road Test in Chapter Six.

If no a priori knowledge is available, then in the proposed local approximation algorithm, the parametric family $\Theta$ should be chosen to be a subset of the collection $L_\Theta$ with locally convergent functions. A parametric function is locally convergent if, for any point $x$ in the domain of a given continuous function, it can approximate in measure the given continuous function on a sufficiently small subset of nearby points of $x$ including the point $x$. Constant functions and functions with a parameterized constant term are locally convergent. Complete polynomials and trigonometric functions are some simple functions that satisfy the local convergence criterion. We say a parametric function is complete if it can perfectly fit any data set whose data points are separate from one another and containing the same number of data as the number of parameters in the parametric function.

By increasing the complexity of the subdomain approximation $\theta$, the number of developed subdomains might be reduced, however the fitting (training) in each subdomain becomes more complicated. If the input domain $\mathcal{D}$ is not partitioned ($s=1$) and $\theta$ is chosen to be a complex multivariate function with unknown parameters then the proposed local approximation becomes a traditional, global, parametric approximation. In this case the proposed method has no advantage in comparison with parametric methods including mapping neural networks with ad hoc architectures. The power of a local approximation is in domain reduction because any complex function can be approximated by locally convergent, simple functions if the region of the approximation is small enough. By partitioning the input domain to subdomains, we also cluster the training data to smaller groups each belonging to some subdomain. Therefore we divide the problem of data fitting (training) to smaller problems which can be solved easier and faster than the original problem. The proposed local approximation algorithm is based on a divide-and-conquer strategy that has a faster training process than global methods.

Since the subdomain approximation function $\theta$ is a locally convergent function and the random partitioning scheme, explained in Section 2.3, can develop subdomains which are as small as locally required as the amount of data increases, therefore the local approximation $F$ constructed by the proposed HARP algorithm converges to $G$ in some measure such as the $L_2$ norm. In other words, the constructed approximation is consistent. One should be aware that not all parametric families $\Theta$ can be combined with all partitioning schemes to develop a consistent, local approximation method. The local convergence of $\Theta$ and subdomain shrinking characteristic of the partitioning scheme are two necessary requirements for the consistency of a local approximation. On the other hand, there are many locally convergent parametric families and subdo-
main shrinking partitioning procedures that their combinations can establish consistent approximation methods. In this study, we unify a large number of these methods that are $L_p$ compatible, in a general formulation, or to be more logically correct, in a general approximation philosophy.

A parametric approximation process is needed to fit the data in each subdomain developed by the HARP algorithm. Since the parametric estimation process for $\theta(x; w)$ (subdomain training) is the most costly part of the HARP algorithm, it is best to simplify and speed up this operation when no a priori knowledge dictates the structure of $\theta$. Hence, the subdomain approximation $\theta$ is chosen to be simple and linear with respect to its parameters. In general $\theta$ can be chosen from the following class of functions

$$\theta(x; w) = \sum_{j=1}^{M} w_j \phi_j(x)$$

(2.5)

where the basis functions $\phi_j$ are simple functions such as polynomials, exponential, or trigonometric functions. If the basis functions $\phi_j$ belong to the collection $L_p$, the subdomain approximation function in Eqn. (2.5) also belongs to $L_p$. For example, if the basis functions $\phi_j$ are polynomials then the expanded form of $\theta$ can be written as

$$\theta(x; w) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i}^{n} w_{ij} x_i x_j + ...$$

(2.6)

and constructed using scalar summation and multiplication of univariate functions $\{x_i\}_{i=1}^{n}$. Therefore the function in Eqn. (2.6) with finite parameters belongs to the $L_p$ space.

The subdomain approximation $\theta(x; w)$ fits the data in a given subdomain by adjusting the parameters $w$. Parameter estimation (training) is generally accomplished through a minimization process. The parameters $w$ are computed by minimizing a fitness index $E(w)$ that is defined as a function of the residual vector $e$. The $k$th residual $e_k$ is the difference between the computed output value $\theta(x_k; w)$ and the target output value $y_k$ for the $k$th training data point $x_k$ in the given subdomain; i.e., $e_k = y_k - \theta(x_k; w)$. The subdomain data fitting (training) of the HARP algorithm is defined by the following parameter estimation problem

$$\minimize_{w} E(w) = E[e(w)]$$

(2.7)

The form of the fitness index $E(w)$ defines the kind of parameter estimator is used and the minimization algorithm used to solve problem (2.7) sets the parameter estimation process. One can find a rich literature about parametric modeling in research fields like statistics, system identification, control, information theory, machine learning, filtering, pattern recognition, prediction, and simulation. There are many variations of parametric modeling techniques using different combinations of existing parameter estimators and parameter estimation processes for specific classes of parametric models. One can use any of these parametric methods for the subdomain approximation stage of the HARP algorithm in accord with the nature of one's problem and the capabilities of one's computational tools. Here we briefly review some of the more famous parameter estimators and estimation processes. Also we discuss truncation of the subdomain approximation for subdo-
mains with a small number of data points and finally we explain the subdomain approximation procedure used to obtain results represented in this dissertation.

2.2.1 Parameter Estimators

Two main classes of parameter estimators that are suitable for mapping approximations are M and R estimators. An M-estimator is a maximum likelihood type and an R-estimator is based on ranks. In general, the fitness index for an M-estimator is defined for the subdomain parametric approximation of the HARP algorithm as follows

\[ E(w) = \sum_{k=1}^{m} \tau[e_k(w)] \]  

(2.8)

where \( m \) is the number of data in a given subdomain and the function \( \tau \) is related to the likelihood function for an appropriate choice of the residual distribution \( q(e_k) \); i.e., \( \tau(e_k) = -\ln q(e_k) \). For example, for the method of least squares (that is an M-estimator), the function \( \tau(z) \) is equal to \( \frac{1}{2}z^2 \) and corresponds to a normal distribution of the residuals. Also, the method of least absolute values corresponds to \( \tau(z) = |z| \) and a double exponential distribution of residuals. One can build a variety of M-estimators assuming different residuals distributions.

The method of least squares is certainly the most popular and most widely applied class of estimators. However, it is well known that when the residuals do not have a normal distribution, particularly those with tails heavier than the normal, the extremes have a large influence on the least squares estimators. The outliers generated by the heavy-tailed distributions pulls the least squares estimates too much toward themselves. Consequently, an examination of residuals cannot identify these outliers because their residuals have been made artificially small. An important class of estimators called robust estimators have been created to reduce the effect of outliers on the final estimates. A robust estimator tends to be insensitive to outliers and leave their residuals large (Andrews, et al. 1972, Andrews 1974, Hill and Holland 1977, Hogg 1974, Huber 1972 and 1981, Launer and Wilkinson 1979).

One of the most satisfying class of robust estimators comes from modifying M-estimators. A robust M-estimator weighs large residuals less heavily than small residuals by modifying the \( \tau \) function in Eqn. (2.8). The weight given to each residual in the first-order-necessary-condition system of equations for problem (2.7) is determined by the derivative of the \( \tau \) function, often called the influence function. The influence function for least squares is unbounded, and thus least squares tends not to be robust when outliers are present in the data. For robust M-estimators, the influence function is bounded and generally is small for large residuals.

For example, the Huber’s robust function is defined as follows:

\[
\tau(z) = \begin{cases} 
\frac{1}{2}z^2 & |z| \leq c \\
-c^2 & |z| > c 
\end{cases}
\]  

(2.9)

where good choices for \( c \) are between 1 and 2. Other robust fitness indices like the Ramsay function, Andrew’s wave function, the Hampel function, and Tukey’s biweight function can be found in the literature (Launer and Wilkinson 1979).
The robust fitness indices are not convex and their corresponding parameter estimation problems should be solved by iterative, minimization algorithms with robust convergence. The robust estimators are extremely helpful in locating outliers. It is recommended to use both a robust and a classical estimator for a data analysis. If the results of both procedures are not in substantial agreement, then reasons for the difference should be identified. The data that are down-weighted in the robust fit should be carefully examined as possible outliers.

The M-estimators, robust or classical, are the most popular and computationally efficient estimators. Another class of robust estimators are the R-estimators. R-estimation is a procedure based on ranks. For problem (2.7), the fitness index of an R-estimator can generally be written as

$$E(w) = \sum_{k=1}^{m} e_k(w) \zeta(R_k)$$

(2.10)

where \(\zeta\) is a score function whose domain is the set of ranks \(\{R_k\}_{k=1}^{m}\) which are integers from 1 to \(m\) and \(R_k\) is the rank of the \(k\)th residual \(e_k\). Two famous score functions are Wilcoxon scores where \(\zeta_i = i\) and the median scores where \(\zeta(i) = 1\) if \(i \leq (m+1)/2\) and \(\zeta(i) = 1\) if \(i > (m+1)/2\) (Adichie 1967, Hogg and Randles 1975, Jaeckel 1972, Jureckova 1977). The R-estimates are more difficult to obtain computationally than M-estimates. Furthermore, under certain conditions R-estimators are asymptotically equivalent to M-estimators (Jureckova 1977). One can also use the order statistics of absolute or squared residuals like their median, supremum or weighted summation of percentiles as the fitness index \(E(w)\).

The fitness index \(E(w)\) can be augmented with penalty terms to reduce some of the defects in the constructed parametric mapping \(\theta(x; w)\), such as the growth of parameters \(w\) or lack of smoothness. These augmented parameter estimators hope for better generalization by not adapting as perfectly to the training data. To limit the growth of parameters, for example, the augmented fitness index can be written as:

$$E(w) = E[e(w)] + \beta \sum_{k=1}^{M} w_k^2$$

(2.11)

where \(M\) is the number of parameters \(w\) in the parametric mapping \(\theta(x; w)\) and \(\beta\) is a penalty parameter. The fitness index \(E(w)\) in Eqn. (2.11) is the residual-based fitness index \(E[e(w)]\) that is penalized by the norm of parameters vector \(w\). Minimization of \(E(w)\) in Eqn. (2.11) represents a trade off between reducing residuals and growing parameter values. When \(E[e(w)]\) is the sum of squared residuals, minimizing \(E(w)\) in Eqn. (2.11) leads to the ridge estimator that is more stable than the least squares estimators for collinear data (Sen and Srivastava 1990).

A smoothing-based, penalized fitness index is another popular augmented fitness index that is used for smoothing splines and the method of regularization (Eubank 1988). The smoothing-based fitness index can be written as

$$E(w) = E[e(w)] + \beta \int_{\mathbb{B}} [P\theta(x; w)]^2 dx$$

(2.12)

where \(\mathbb{B}\) is the support of the parametric function \(\theta(x; w)\), \(\beta\) is a smoothing, penalty parameter, and \(P\) is a differential operator. A common choice for \(P\) is the second-derivative operator. The main reason for using
the smoothing, penalty term defined in Eqn. (2.12) is to control the smoothness of the constructed mapping which is represented by its higher derivatives. The smoothness control prevents a residual-based estimator from adjusting the parameters such that the constructed mapping has small residuals at training data points but is not smooth. In other words, the smoothing, penalized fitness index (2.12) prevents developing unnecessary undulations on the surface defined by the functional form of the constructed mapping $\theta(x; w)$. Unnecessary undulation decreases smoothness and generalization without significantly decreasing the residuals.

The common technique for determining penalty parameters in the augmented fitness indices (2.11) and (2.12) is the method of cross-validation (Eubank 1988). We will explain the cross-validation method for performance estimation in Chapter Five.

### 2.2.2 Parameter Estimation Algorithms

A parameter estimation algorithm is a minimization procedure for solving the parameter estimation problem (2.7). The estimation process is sometimes referred to as the learning or training process. According to the physics of the problem (2.7), the parametric function $\theta(x; w)$ can be interpreted as a system model, inference machine, parametric approximation, neural network, or coding machine. Respectively, one can find many parameter estimation schemes in the research areas of system identification and control, machine learning, regression analysis, neurocomputing, and information theory. In this section, we look at the parameter estimation process from the optimization viewpoint.

The first order necessary conditions for the optimization problem leads to a system of equations as follows

$$\frac{\partial E[e(w)]}{\partial w_k} = 0 \quad k = 1, ..., M$$

(2.13)

where $M$ is the number of parameters in the parametric function $\theta(x; w)$. In general, the system of equations (2.13) is nonlinear with respect to parameters $w$ and should be solved iteratively. Regarding the required accuracy and computational efficiency, a suitable estimation algorithm can be selected from many existing algorithms like steepest descent, conjugate gradient, recursive quadratic programming, and quasi-Newton methods for local minimization and random search methods, genetic algorithms, random sampling methods, and simulated annealing algorithms for global minimization (Luenberger 1989, Tom and Zilinskas 1989, Horst and Tuy 1990).

The nonlinearity of Eqn. (2.13) depends on the form of the fitness index $E$ and the structure of the parametric function $\theta(x; w)$. When $E$ is the sum of squared residuals and $\theta(x; w)$ belongs to the class of parametric functions that are linear with respect to their parameters and defined in Eqn. (2.11), the subdomain training problem (2.7) takes the form as

$$\min_w E(w) = \sum_{k=1}^{m} \left[ \sum_{j=1}^{M} w_j \phi_j(x_k) - y_k \right]^2$$

(2.14)

whose first order necessary conditions defined in Eqn. (2.13) can be written as follows
where \( y \) is the vector of target values for the training data points in the given subdomain and \( U \) is the \( m \times M \) matrix with components given by \([U]_{jk} = \phi_j(x_k)\). If \( U^T U \) is invertible then the explicit solution for the system of equations (2.14) is

\[
[U^T U] w = U^T y
\]

(2.15)

otherwise \( U^T U \) is rank deficient indicating collinearity among training data points. For collinear data, the classical least squares estimator (2.16) can be replaced by a more stable estimator like singular-value decomposition, principal components, Shrunken, iteration, inversion, Bayesian, and minimax estimators (Trenkler 1981). For example, the iteration estimator is defined as follows

\[
w = [U^T U]^{-1} U^T y
\]

(2.16)

where \( \beta \) is a positive parameter less than twice the inverse of the greatest eigenvalue of \( U^T U \). By increasing \( L \), the estimated value \( w \) in Eqn. (2.17) converges to a unique vector that is equal to the vector in Eqn. (2.16) when the data are not collinear. The iteration estimator (2.17) replaces the matrix inversion in the estimator (2.16) with the matrix multiplication and summation. The recursive formula for the iteration estimator is

\[
w_{j+1} = w_j + \beta U^T (y - Uw_j)
\]

(2.18)

where \( w_j \) is the parameter vector at the \( j \)th iteration. The recursive estimator (2.18) updates parameters after a complete presentation of the training set that is required to generate and store \( U \) and \( y \). This approach is called batch or periodic updating. The recursive estimator (2.18) enables one to follow the history of estimated parameters through the estimation process and whenever all parameters reach their limit values, to stop the process (i.e., the number of iterations \( L \) is large enough). Considering the definitions for the fitness index \( E(w) \), given in Eqn. (2.14), and the matrix \( U \), the recursive estimator (2.18) can be defined in a more general form as follows

\[
w_{j+1} = w_j - \beta \left. \frac{\partial E(w)}{\partial w} \right|_{w=w_j}
\]

(2.19)

which is a standard steepest descent iteration for the nonlinear optimization problem (2.7). In the sense of neural networks, Eqn. (2.19) is the generalized delta rule to train a mapping neural network (Rumelhart, et al. 1986).

Two main classes of parameter estimation algorithms are on-line and off-line algorithms. An off-line algorithm treats the data as a complete block of information that is available prior to analysis. In contrast to an off-line algorithm, an on-line algorithm deals with sequential data and recursively updates parameters within the time limit imposed by the data presentation period. No strict time limit is imposed on an off-line scheme. On the other hand, in many applications, it is necessary to use a relatively simple scheme for on-line
parameter updating to meet the imposed time constraint. The off-line estimation algorithms are more stable and convergent than the on-line algorithms. While the on-line algorithms are more practical for real-time analyses than the off-line ones.

The parameter estimation algorithms previously explained for the subdomain training problem (2.7) are off-line schemes. In this dissertation, we assume the whole data set is available prior to analysis and thus, we use off-line algorithms. A rich literature about on-line estimation algorithms can be found in research areas like control, pattern recognition, signal processing, machine vision, robotics, and speech recognition. Books by Ljung and Soderstrom (1982) and Goodwin and Sin (1984) are good references for on-line estimation algorithms.

An example of an on-line estimation scheme is the sequential least-squares algorithm that is expressed as

$$w_{j+1} = w_j + P_j g_j [y_j - \theta(x_j; w_j)] \quad j \geq 0$$  \hspace{1cm} (2.20)

where the matrix $P$ is updated as follows

$$P_j = P_{j-1} - \frac{P_{j-1} g_j g_j^T P_{j-1}}{1 + g_j^T P_{j-1} g_j} \quad (2.21)$$

and the gradient vector $g_j$ is defined as

$$g_j = \frac{\partial \theta(x_j; w)}{\partial w} \bigg|_{w = w_j} \quad (2.22)$$

where $(x_j, y_j)$ is the training data pair presented at the $j$th iteration. The initial matrix $P_0$ is a positive definite matrix that represents the confidence in the initial parameters $w_0$. The recursive nature of on-line algorithm makes the behavior of the algorithm depend on the pattern of presentation of the training data. The computational process at each iteration is based on the presented data pair at that iteration and does not require the rest of the data.

### 2.2.3 Termination Criterion for the Subdomain Training

In the HARP algorithm, a termination criterion for subdomain training is needed. The termination criterion is a measure of the goodness of the fit developed by the subdomain approximation $\theta(x; w^*)$ constructed by the parameter estimation process. The parameter vector $w^*$ represents the estimated parameters. In the HARP algorithm, a subdomain with acceptable fit is not subject to further training. On the other hand, a subdomain with poor fit is partitioned.

A reasonable choice for a goodness-of-fit measure is the fitness index $E(w^*)$ computed for the estimated parameters. The termination criterion takes the form as follows

$$E(w^*) \leq \varepsilon \quad (2.23)$$
where $\epsilon$ is a tolerance value selected by the experimenter. Also the order statistics of absolute or squared residuals like their mean, medium, or maximum can be used as the termination criterion. For example, the maximum-residual and mean-squared-residual termination criteria are respectively defined as follows:

$$
\max_{1 \leq k \leq m} \{e_k(w^*)\} \leq \epsilon \\
\frac{1}{m} \sum_{k=1}^{m} e_k^2(w^*) \leq \epsilon
$$

(2.24)

Furthermore, one can use a measure for the performance of $\theta(x; w^*)$ on the training or test data points in its subdomain as the termination criterion. In chapter five, we will explain performance measures like the cross-validation, bootstrap, and predicted-squared-error measures. Also, in chapter five, we will develop a procedure for selecting the best tolerance value $\epsilon$.

### 2.2.4 Variable Selection

Theoretical considerations or prior experience can be helpful in selecting the structure of the subdomain parametric approximation $\theta$. However, in most practical problems, we do not have any a priori knowledge about the structure of $\theta$. In these situations, the HARP method recommends selecting $\theta$ from the popular classes of approximations like polynomials, trigonometric functions, mapping neural networks, and radial basis functions. The selected $\theta$ may include functional terms that are not significantly influential in representing the behavior of the data in subdomain $B$ corresponding to $\theta$. For example, $\theta$ might be a second-degree polynomial while the data in $B$ might have essentially linear behavior. Eliminating superfluous terms in $\theta$ decreases the complexity of $\theta$, increases the confidence in its estimated parameters, and increases its generalization. Consequently, the generalization of the approximation constructed by the HARP method increases.

Elimination of unnecessary terms in a parametric model is well-known in the statistical modeling and is called the variable selection problem. A simple variable selection technique is to fit the assumed parametric model $\theta$ to the given data and then eliminate the functional terms in $\theta$ corresponding to parameters with small estimated values. The robustness of this technique is highly sensitive to the amount of noise added to the target values, and is not recommended when the amount of noise is large. More robust techniques for variable selection have been developed in regression analysis. These techniques try to minimize a performance measure by searching through the pool of all possible submodels of the assumed parametric model. The search can be done by growing the model from a simple form to a more complex one, by pruning the model from a complex form to a simpler one, or by stepwise growing and pruning (Montgomery 1982).

In the HARP algorithm, when $\theta$ cannot acceptably fit the data in a subdomain, the subdomain is partitioned. Hence by growing the HARP partitioning tree, the size of subdomains decreases. Consequently, the amount of data in a subdomain decreases. Therefore, the number of data in a subdomain, $m$, may become smaller than the number of parameters, $M$, in the parametric function $\theta(x; w)$. In this case, the estimated parameters $w$ have low confidence and $\theta$ has poor generalization. One way for solving this problem is to use
variable selection techniques. These techniques select a submodel of the assumed parametric model that has good performance with a number of parameters smaller than \( m \). Another way to get around the problem of parametric fitting for small subdomains is to simply downsize the assumed parametric model. In this technique, the subdomain approximation \( \theta \) has the flexibility to reduce its number of parameters whenever the number of data in a subdomain, \( m \), is smaller than its number of parameters. For example, the higher-degree terms in a polynomial \( \theta \) are eliminated, the terms with higher frequencies are truncated in a trigonometric \( \theta \), the number of hidden units are reduced for a neural network \( \theta \), and the number of radial basis functions are reduced.

Different truncation schemes can be used. A truncation scheme may eliminate blocks of functional terms in \( \theta \). For example, a quadratic polynomial is truncated to a linear polynomial or a few hidden units and their corresponding connections are eliminated from a neural network. We refer to this kind of truncation as the **block truncation**. For a block truncation scheme, the number of parameters in the truncated \( \theta \) may become smaller than \( m \). The truncation scheme may be set such that the number of parameters in the truncated \( \theta \) become equal to \( m \). For example, some of the second-degree terms in a quadratic polynomial are eliminated to produce a truncated quadratic polynomial or some of the connections in a neural network are eliminated without reducing the number of hidden units. We refer to this kind of truncation as the **term truncation**. For example, for a linear parametric approximation \( \theta = w_0 + w_1 x_1 + w_2 x_2 \) and \( m = 2 \), the function \( \theta_1 = w_0 \) is a block-truncated function and \( \theta_2 = w_0 + w_1 x_1 \) is a term-truncated function. The numbers of parameters in \( \theta_1 \) and \( \theta_2 \) are less than \( m \) and equal to \( m \), respectively. Both block and term truncation schemes lead to a constant function when \( m \) is equal to one. In a block truncation scheme, a block of functional terms represents a macro-structure of the assumed parametric model \( \theta \) like polynomial terms with the same degree, trigonometric terms with the same frequency, or connections connected to the same hidden unit in a neural network. On the other hand, in a term truncation scheme, we work with the micro-structures like individual polynomial terms or connections.

Truncation schemes are not as sophisticated as the statistical variable selection techniques. However, they are more easily implemented. The use of statistical variable selection techniques or truncation schemes are highly recommended for the subdomain training in the HARP algorithm. We will show in the following sections that the HARP algorithm with a simple truncation scheme is a powerful tool for multivariate mapping approximation.

### 2.2.5 Selected Subdomain Approximation Procedure

In previous sections we have explained different strategies for selecting the structure of the parametric subdomain approximation \( \theta \), selecting the parameter estimator, and estimating the parameters. One can combine these strategies to develop a subdomain approximation process that is suitable for his problem. The nature of the data fitting problem generally dictates what kind of structure for \( \theta \), estimator, and estimation algorithm should be used. The parametric function \( \theta \) should be locally convergent and belong to \( L_\infty \). The function \( \theta \) can be selected from function spaces like polynomials, splines, radial basis functions, and mapping neural networks. The number of parameters in \( \theta \) can be reduced using variable selection techniques or
truncation schemes. When the data contains outliers, robust estimators can be used. For real-time analyses, on-line parameter estimation algorithms can be used.

In this dissertation, we work with simulated data or actual data that are devoid of outliers. Also the whole data set is available before the analysis. Therefore, we use the classical least-squares estimator with off-line parameter estimation algorithms. The maximum residual for the training data points, as defined in Eqn. (2.24), is used as the termination criterion. When the structure of the subdomain approximation $\theta$ is not selected by a priori knowledge, we chose $\theta$ to be linear with respect to its parameters meaning it belongs to the class of functions shown in Eqn. (2.5). The basis functions are simple functions like polynomials, exponential, or trigonometric functions. Hence, the subdomain training problem is the same as Eqn. (2.14). We compute the parameters using the explicit solution (2.16). When the data are collinear, we use the singular-value decomposition to solve the matrix equation (2.15). A term truncation scheme is used to deal with the training process of a subdomain with a small number of data.

We do not use any penalty term with the fitness index to control the smoothness of the subdomain approximations, like Eqn. (2.12), and to prevent the growth of their parameters, like Eqn. (2.11). However in Section 2.3, we will explain how to smooth the mismatching of the local approximations in the neighboring subdomains. Furthermore, we will show that the superposition of a sample of approximations built by the HARP algorithm reduces the localized disturbances among the subdomains and controls the global smoothness of the constructed mapping. The growth of estimated parameters $w_\beta^*$ in a subdomain approximation $\theta_\beta(x; w_\beta^*)$ can happen especially when the data in the subdomain $\mathcal{B}$ are collinear. Although for collinear data, the singular-value decomposition increases the numerical robustness of the training process and mitigates the defects of unrealistically large parameters, we use a squashing function $\Phi$ to bound the outputs of a parametric function $\theta_\beta(x; w_\beta^*)$. The squashing function $\Phi$ is a ramp function defined as follows

$$\Phi(z; a, b, \mu) = \begin{cases} a - \mu(b - a) & z < a - \mu(b - a) \\ z & a - \mu(b - a) \leq z \leq b + \mu(b - a) \\ b + \mu(b - a) & z > b + \mu(b - a) \end{cases}$$

and is shown in Fig. 2.2. The free parameter $\mu \geq 0$ controls the width of the ramp region of the squashing function $\Phi$ and the parameters $a$ and $b$, where $b > a$, controls the lower and upper bounds of $\Phi$, respectively.

For a point $s$ in the subdomain $\mathcal{B}$, the predicted output $\theta_\beta(s; w_\beta^*)$ is bounded using the squashing function $\Phi$ and takes the value $\Phi(\theta_\beta(s; w_\beta^*); l_\beta, u_\beta, \mu)$ where $l_\beta$ and $u_\beta$ are

$$l_\beta = \min\{\theta_\beta(x; w_\beta^*): x \in \mathcal{B} \cap \mathcal{I}\}$$

$$u_\beta = \max\{\theta_\beta(x; w_\beta^*): x \in \mathcal{B} \cap \mathcal{I}\}$$

The values $l_\beta$ and $u_\beta$ are respectively the minimum and maximum of the outputs predicted by the subdomain approximation $\theta_\beta(x; w_\beta^*)$ for the training data points in the subdomain $\mathcal{B}$.
In the training process of the HARP algorithm, the parameters \( l_{\mathcal{B}} \) and \( u_{\mathcal{B}} \) of the squashing function \( \Phi \) corresponding to the subdomain \( \mathcal{B} \) are simply computed using Eqn. (2.26) after the parameters \( w_{\mathcal{B}}^* \) are estimated. The free parameter \( \mu \) is a preassumed parameter and does not change during the training process. Hence, all the parameters of the squashing function \( \Phi \) are computed explicitly and are not determined, like the parameters \( w_{\mathcal{B}}^* \), through a parameter estimation (training) process. The free parameter \( \mu \) controls the range of the squashed subdomain approximation \( \Phi(\theta_{\mathcal{B}}) \). For \( \mu = \infty \), the squashing function \( \Phi \) is an identity function and the squashed function \( \Phi(\theta_{\mathcal{B}}) \) is equal to \( \theta_{\mathcal{B}} \) and the range of \( \Phi(\theta_{\mathcal{B}}) \) is equal to the range of \( \theta_{\mathcal{B}} \) that may be unrealistically large. For \( \mu = 0 \), the range of the squashed function \( \Phi(\theta_{\mathcal{B}}) \) is bounded and is equal to \([l_{\mathcal{B}}, u_{\mathcal{B}}]\). To set the free parameter \( \mu \), one should start with a big \( \mu \) value and check the predicted outputs for points between training data points. If no unnaturally large, predicted output is observed, then the chosen \( \mu \) value is good otherwise decrease the \( \mu \) value. A conservative \( \mu \) value is zero that does not allow the predicted output for any point in the input domain to be out of range of training target values. We use \( \mu = 0 \) for all the analysis in this dissertation.

If the subdomain approximation \( \theta \) belongs to the collection \( \mathcal{L}_{\mathcal{B}} \), then the squashed function \( \Phi(\theta) \) belongs to \( \mathcal{L}_{\mathcal{B}} \) because it is a composition of a univariate function \( \Phi \) on an \( \mathcal{L}_{\mathcal{B}} \) function \( \theta \). The squashing function defined in Eqn. (2.25) can be replaced with any bounded, continuous univariate function if the selected function is an identity function for the interval \([a, b]\) described in Eqn. (2.25). This restriction for a squashing function guarantees that for the training data points, the outputs predicted by a squashed subdomain approximation \( \Phi(\theta) \) are equal to the outputs predicted by the subdomain approximation \( \theta \).

### 2.3 Subdomain Partitioning

In the proposed HARP method, a subdomain with a poor fit is partitioned. The main challenge in establishing an algorithm for data partitioning is the constraint that a subdomain characteristic function must belong to the collection \( \mathcal{L}_{\mathcal{B}} \). To meet this constraint, the partitioning must be accomplished with the \( \mathcal{L}_{\mathcal{B}} \) fundamental operations of addition, multiplication and composition of univariate functions. The key is a suitable description of the characteristic functions for the subdomains. To build the characteristic function we introduce the concept of the partitioning function \( \psi(x) \) and its inverse image.

To describe the partitioning process we will consider the partitioning of a single subdomain. The complete partitioning algorithm will then be deduced by recursion. Assume that \( \mathcal{B} \) is a subdomain of the input...
domain that has developed at some intermediate stage of the HARP algorithm. Also assume that there are \( m \) data points in \( \mathcal{B} \) and that \( \mathcal{B} \) requires further partitioning due to a poor function fit in that subdomain. The essential idea is as follows: A partitioning function \( \psi(x) : \mathbb{R}^n \rightarrow \mathbb{R} \) is introduced to map the multidimensional data to the real line. The real line is then partitioned into intervals. The inverse images of the partition intervals constitute a partitioning of the original domain. The characteristic function is then simply constructed from the partitioning function.

Consider a measurable, nonconstant function \( \psi(x) : \mathbb{R}^n \rightarrow \mathbb{R} \). The values of this function at the data points are a finite set \( \mathcal{Z} \) of real numbers that, after sorting into ascending numerical order, take the form

\[
\mathcal{Z} = \{ z_i = \psi(x) : x \in \mathcal{B} \cap \mathcal{T} \, , \, z_i \in \mathbb{R} \, , \, z_i \leq z_{i+1}, \, i = 1, \ldots, m' \} \tag{2.27}
\]

The partitioning function \( \psi \) behaves as a projection and might map some data points \( x \) to the same point on the real line. Therefore, \( \mathcal{Z} \) is a set containing \( m' \leq m \) discrete points. We wish to cover \( \mathbb{R} \) with \( r \) disjoint intervals that contain the discrete points \( \{ z_i \}_{i=1}^{m'} \), with each interval containing at least one projected data point. We denote the bounds of the intervals with the set of end points \( \{ d_0, d_1, \ldots, d_r \} \) where \( d_0 = -\infty \) and \( d_r = +\infty \). The \( j \)th interval will be designated as \( I_j \) and is described in terms of its endpoints as

\[
I_j = \{ z : d_{j-1} < z \leq d_j \} \quad \text{for } j = 1, \ldots, r \tag{2.28}
\]

These intervals constitute a partition of the real line, designated as \( \mathcal{C} \), with the property

\[
\mathcal{C} = \{ I_j : \bigcup_{j=1}^{r} I_j = \mathbb{R} , \, I_k \cap I_j = \emptyset \, \forall k \neq j \} \tag{2.29}
\]

A typical partition is shown schematically in Fig. 2.3.

The partitioning function \( \psi \) should be chosen such that the members of the set \( \mathcal{Z} \) are distinct so that the intervals of the partition \( \mathcal{C} \) can be adequately defined. For any multivariate function \( \psi(x) : \mathbb{R}^n \rightarrow \mathbb{R} \) the inverse image of an interval \( [a, b] \) in its range is defined by

\[
\psi^{-1}[a, b] = \{ x : x \in \mathbb{R}^n , \, a \leq \psi(x) \leq b \} \tag{2.30}
\]

The inverse image is a set in \( \mathbb{R}^n \). The example partitioning function \( \psi(x_1, x_2) = x_2 - x_1 \) is shown in Fig. 2.4. Points below the diagonal line are mapped to the interval \( I_1 \) (negative numbers) and points above the diagonal line are mapped to the interval \( I_r \) (positive numbers). Points on the diagonal are mapped to the interval \( I_2 \) (zero)

![Fig. 2.3 Partitioning the range of \( \psi(x) \)](image)

26
are mapped to the interval $I_2$ (positive numbers). The inverse images of the intervals $I_1$ and $I_2$ split $\mathcal{B}^2$ into two subdomains along the diagonal $x_1 = x_2$.

Define the univariate gate function $\Gamma$ as

$$\Gamma(z; a, b) = \begin{cases} 1 & \text{if } a \leq z \leq b \\ 0 & \text{otherwise} \end{cases}$$

This function is shown schematically in Fig. 2.5. The characteristic function of the inverse image of the interval $[a, b]$ under $\psi$ is then computed by

$$\chi(x) \mid_{\psi^{-1}[a,b]} = \Gamma(\psi(x); a, b)$$

If $\psi$ belongs to the collection $\mathcal{L}_\mathcal{B}$, then the proposed form for $\chi$ in Eqn. (2.32) is the composition of a univariate function $\Gamma$ on an $\mathcal{L}_\mathcal{B}$ function $\psi$ and therefore, belongs to $\mathcal{L}_\mathcal{B}$.

The inverse image of the intervals $\{I_1, I_2, ..., I_r\}$ partition $\mathcal{B}^n$ into $r$ measurable sets. These sets must be restricted to the subdomain $\mathcal{B}$. Let $\mathcal{B}_j$ be the intersection of $\mathcal{B}$ and the inverse image of the $j$th interval, $\mathcal{B}_j = \psi^{-1}(I_j) \cap \mathcal{B}$. The characteristic function of $\mathcal{B}_j$ is simply the product of the characteristic functions of the parent subdomain $\mathcal{B}$ and the inverse image of the $j$th interval:

$$\chi_{\mathcal{B}_j} = \chi_{\mathcal{B}} \cdot \chi_{\psi^{-1}(I_j)} = \chi_{\mathcal{B}} \cdot \Gamma(\psi(x); \; d_{j-1}, d_j)$$

$$\Gamma(z; a, b)$$

Fig. 2.5 Gate function
During the HARP algorithm, the input domain $\mathcal{D}$ is partitioned to subdomains by using the concept of inverse image and the characteristic function of a subdomain is recursively built from the characteristic function of its parent subdomain and its corresponding inverse image, according to Eqn. (2.33). In the following sections, we explain how to select the partitioning function $\psi$, the interval bounds $\{d_i\}$, and the number of splits $r$. Also, we will discuss how one can smooth the mismatching of the local approximations in neighboring subdomains by modifying the characteristic function $\chi$.

2.3.1 Selection of the Partitioning Function

The behavior of the approximation constructed by the HARP algorithm is controlled by the local approximations on the subdomains. Therefore, the subdomains manage how information from different regions of the input domain affects the approximation. To keep remote data points from influencing the approximation locally, each subdomain should be a connected region with a small aspect ratio. The shape and size of the generated subdomains depend on the partitioning function $\psi$. For the proposed HARP algorithm, we use a parametric $\psi$ function that is defined in terms of a finite number of unknown parameters $c$. The form of the partitioning function $\psi$ is presumed and does not change during the training process. The parameters $c$ are chosen randomly with the constraint that $\psi$ does not become constant for all of the data points in a corresponding subdomain. In other words, the set $\mathcal{Z}$ in Eqn. (2.27) does not become a singleton. This restriction assures the existence of distinct projected points in the set $\mathcal{Z}$ that can be used to partition the subdomain and its corresponding data. Because the parameters $c$ are determined by a random number generator, no estimation (training) process is needed to obtain these parameters and they are determined with simple computations. Unlike classification and regression trees that search for the best split of a subdomain through a minimization process of a splitting index like misclassification or impurity indices (Breiman, et al. 1984) with respect to the partitioning parameters $c$, in the HARP algorithm, a subdomain can be partitioned in an infinite number of ways because of the randomness of $c$. In this dissertation, we will show that the random partitioning not only speeds up the training process of the HARP algorithm but also introduces a new philosophy for the data-based approximation of multivariate mappings.

A simple form for the partitioning function $\psi$ is

$$\psi(x; c) = \sum_{i=1}^{n} c_i x_i$$

(2.34)

where $n$ is the dimension of the input domain $\mathcal{D}$ and $c$ is the vector of random parameters. The partitioning function defined in Eqn. (2.34) is a linear polynomial and belongs to the collection $\mathcal{L}_\mathcal{D}$. The splitting surface $\psi(x) = \text{constant}$ has the equation of a hyperplane. Thus, the inverse image of $\psi$ for any interval in $\mathcal{H}$ has straight boundaries and is the intersection of two half spaces. One can prove that a convex region partitioned by a hyperplane gives convex subregion. Also a convex region is connected. Because, the linear partitioning function partitions a convex input domain into connected convex subdomains, it is a good choice for a local approximation. For a non-convex input domain, one can first partition the domain into convex subdomains and then uses the HARP algorithm with the linear $\psi$ to fit the data in each convex subdomain.
For the results in this dissertation, we use the linear partitioning function defined in Eqn. (2.34) for the subdomain partitioning of the HARP algorithm. Furthermore, the random parameters $c$ of the linear $\psi$ function are independent from one another and have a standard normal distribution. The parameters $c$ are determined by a random number generator for normally distributed deviates (Press, et al. 1990).

### 2.3.2 Selection of the Splitting Thresholds

We refer to the bounds of the intervals, $\{d_i\}_{i=0}^r$, in the partition $C$, defined in Eqn. (2.29), as the splitting thresholds. The parameters $c$ of the partitioning function $\psi$ controls the alignment of the subdomain splitting surfaces; i.e., $\{\psi(x; c)=d_i\}_{i=1}^{r-1}$, and the splitting thresholds determine the position of these surfaces. In the HARP algorithm, the splitting thresholds are chosen with the constraint that each interval in the partition $C$ contains at least one training data point. This restriction ensures that none of the subdomains in the final partition $C$ of the input domain are empty of training data points.

Let $\mathcal{V}$ be the collection of all subsets of the input domain $\mathcal{D}$ that can be described as the solution set to a system of finite number of inequalities of the form $\psi(x; c) \leq d$ or $\psi(x; c) < d$ where $\psi$ is the HARP partitioning function and parameters $c$ and $d$ are real numbers. All subdomains in the intermediate partitions $\{\mathcal{C}^v\}_{v=1}^{\text{max}}$ and the final partition $\mathcal{C}^\text{max}=C$, as shown in Fig. 2.6, are produced by a finite number of splits described by the inequalities mentioned in the definition of $\mathcal{V}$. Consequently, these subdomains belong to $\mathcal{V}$. For example, as shown in Fig. 2.6, the subdomain number 5 in the intermediate partition $C^2$ is developed by two splits of the input domain, splits number 1 and 2, and the subdomain number 12 in the final partition $C$ is produced by four splits of the input domain, split numbers 1, 2, 4, and 6.

Let $\mathcal{P}_\psi$ be the collection of all partitions of $\mathcal{D}$ whose subdomains belong to $\mathcal{V}$. The collection $\mathcal{P}_\psi$ contains a subset of all partitions of $\mathcal{D}$. A partition $C$ developed by the HARP algorithm belongs to $\mathcal{P}_\psi$. In other words, the collection of partitions developed by HARP, $\mathcal{P}_\mathcal{H}$ is a subset of $\mathcal{P}_\psi$. The collection $\mathcal{P}_\mathcal{H}$ is smaller than $\mathcal{P}_\psi$ because the subdomains in the HARP partitions are restricted to follow the hierarchical and adaptive partitioning scheme of HARP. The subdomains developed by HARP should contain at least one training data point. Therefore, the distribution of subdomains in a HARP partition is controlled by the distribution of train-

![Diagram](image)

Fig. 2.6 Indexing of the splits and partitioned subdomains
ing data points. Also, in the HARP algorithm, a subdomain is partitioned until its corresponding fit is acceptable. In other words, the partitioning scheme of HARP is approximation-error-driven. Consequently, the distribution of subdomains in a HARP partition adaptively follows the complexity of the actual mapping. The generated subdomains are more concentrated in the regions of $\mathcal{D}$ where the actual mapping has complex behavior. Hence, $\mathcal{P}_H$ is a specific subset of $\mathcal{P}_\psi$ whose partitions follow the distribution of the training data points and the complexity of the actual mapping. Although the HARP algorithm randomly partitions the input domain $\mathcal{D}$, it attempts to reasonably distribute subdomains in $\mathcal{D}$ by concentrating them in the regions with complex data behavior and by generating non-empty subdomains.

The scheme for selecting splitting thresholds influences the size and shape of subdomains and consequently the partitions developed by HARP. We use three simple schemes for selecting splitting thresholds. We call these schemes by their corresponding collections of partitions: $\mathcal{P}_H^1, \mathcal{P}_H^2, \text{and } \mathcal{P}_H^3$ schemes.

The $\mathcal{P}_H^1$ scheme selects the splitting thresholds such that each interval in the partition $\mathcal{C}$ defined in Eqn. (2.29) contains an equal number of training data points. The $\mathcal{P}_H^2$ scheme balances the numbers of training data points in offspring subdomains. For partitioning a subdomain to $r$ offspring subdomains, the splitting thresholds are determined as follows

$$d_i = \frac{1}{2}(z_{m'_{i+1}} + z_{m'_i})$$  \hspace{1cm} i = 1, \ldots, r-1 \tag{2.35}$$

where $m'$ is the number of projected data points in the set $\mathcal{Z}$, defined in Eqn. (2.27), and $z_k$ is the $k$th projected data point in $\mathcal{Z}$. The $\mathcal{P}_H^1$ scheme chooses the splitting thresholds with the constraint that each subdomain contains a minimum number of training data points so that there is a sufficient data in each subdomain for a robust fit of the approximation function $\theta$. The minimum number of data points should be greater than the number of parameters in $\theta$. When the amount of data in a subdomain is not enough to assign sufficient data to offspring subdomains, one may stop the training process for this subdomain or partition the subdomain to two subdomains and use a truncation scheme to reduce the number of parameters in $\theta$ for the offspring subdomains.

The $\mathcal{P}_H^2$ scheme randomly selects the splitting thresholds in the interval $(z_1, z_{m'})$. The values $z_1$ and $z_{m'}$ are, respectively, the lower and upper bounds of the projected data points in the set $\mathcal{Z}$. A uniform random number generator with range $(z_1, z_{m'})$ is used to choose the splitting thresholds $\{d_i\}_{i=1}^{r-1}$. The $\mathcal{P}_H^3$ scheme also selects the splitting thresholds randomly but in the discrete set of midpoints for the projected data points; i.e., $\{\frac{1}{2}(z_i + z_{i+1})\}_{i=1}^{m'-1}$. Both $\mathcal{P}_H^1$ and $\mathcal{P}_H^3$ schemes choose the splitting thresholds with the constraints that each interval in the partition $\mathcal{C}$, defined in Eqn. (2.29), contains at least one projected data point. Figure 2.7 shows splitting thresholds selected by $\mathcal{P}_H^1$, $\mathcal{P}_H^2$, and $\mathcal{P}_H^3$ schemes to partition a subdomain into three offspring subdomains. Based on the definitions of these different schemes, this hierarchical relationship can be concluded between their corresponding collections: $\mathcal{P}_H^1 \subset \mathcal{P}_H^2 \subset \mathcal{P}_H^3 \subset \mathcal{P}_\psi$.

### 2.3.3 Number of Splits in a Subdomain

In the HARP algorithm, a subdomain with a poor fit is partitioned into $r$ offspring subdomains. The number of splits in a subdomain, $r$, is a fixed integer selected by the experimenter and is greater than one. The
number of offspring subdomains is equal to the number of intervals in the partition $C$. Since the schemes for selecting the bounds of these intervals are based on the projected data points, the number of intervals, $r$, cannot be greater than the number of projected data points, $m'$, in order to develop non-empty intervals. In the HARP algorithm, if $m'$ is greater than the selected value for $r$, the number of splits is set to be two.

The larger the selected $r$ value is, the faster the size of the subdomains decreases during the training process of HARP and consequently, the faster the training process is. The speed of the subdomain training process of HARP is directly related to the amount of training data in a subdomain. The number of training data points in a subdomain is a function of the size of the subdomain. Furthermore for a selected, big $r$ value the sizes of subdomains in the intermediary partitions of HARP are smaller than for a selected, small $r$ value. Therefore, the training process for these subdomains becomes faster and consequently, the training process of HARP speeds up. The disadvantage of selecting a big $r$ value is that we reduce the chance of developing large subdomains with good fit. Therefore, the number of subdomains in the final partition of HARP increases leading to construction of approximations with large number of parameters. The choice $r=2$ develops the simplest approximation with high probability and is recommended whenever the speed of training is not important.

In the HARP algorithm, for a selected $r$ value greater than two, a subdomain is partitioned through a hierarchy of binary splits. For example, as shown in Fig. 2.8, a subdomain $B$ is partitioned to four offspring subdomains using two different schemes: (a) three parallel splits with different splitting thresholds and (b) a hierarchy of three binary splits. It is evident that a random splitting scheme using a hierarchy of binary splits develops subdomains with smaller aspect ratio than a scheme using parallel splits. Subdomains with small

![Fig. 2.7 Schemes for selecting splitting thresholds](image)

![Fig. 2.8 Splitting schemes](image)
aspect ratios are more desirable because they reduce the influence of remote data points on a local approximation and consequently increase the generalization. Hence, the scheme using the hierarchy of binary splits is better than the scheme using parallel splits. Both splitting schemes require the same number of splitting thresholds but a hierarchy of binary splits needs more parameters $c$ to define its splitting surfaces $\psi(x; c)$. Therefore, a hierarchy of binary splits requires more parameters. However it is recommended because of producing partitions whose corresponding HARP approximations have better generalization.

2.3.4 Boundary Continuity

Along the boundaries between subdomains, the constructed HARP approximation is generally discontinuous because of mismatching of the subdomain approximations $\theta$ in neighboring subdomains, even when the estimated approximations $\hat{\theta}$ are smooth within subdomains. These regions of discontinuity cover a small portion of the input domain. The boundary mismatching is especially important when the number of training data points is small. The subdomain approximations come closer to one another at the interior boundaries as the number of data points increases and the constructed HARP approximation becomes smooth.

The discontinuity of the gate function $\Gamma$, defined in Eqn. (2.31), at the boundaries of the intervals in the partition $C$ leads to building a characteristic function $X$, defined in Eqn. (2.4), that is discontinuous on the boundary of a subdomain. The discontinuity of $X$ causes the mismatching at the interior boundaries of the subdomains. Continuity of the constructed HARP approximation, defined in Eqn. (2.3), can be enforced by modifying the gate function $\Gamma$. The idea is to mollify the gate function so that it is continuous to the higher order derivatives. Define the smooth gate function $\tilde{\Gamma}$, shown schematically in Fig. 2.9, for the $j$th interval $I_j$ in the partition $C$ as follows

\[
\begin{cases}
\tilde{\Gamma}_j(z) = 0 & \text{if } z \leq d_{j-1} - \tilde{r}_{j-1} \\
0 < \tilde{\Gamma}_j(z) < 1 & \text{if } d_{j-1} - \tilde{r}_{j-1} < z < d_{j-1} + t_j \\
\tilde{\Gamma}_j(z) = 1 & \text{if } d_{j-1} + t_j \leq z \leq d_j - \tilde{r}_j \\
0 < \tilde{\Gamma}_j(z) < 1 & \text{if } d_j - \tilde{r}_j < z < d_j + t_{j+1} \\
\tilde{\Gamma}_j(z) = 0 & \text{if } d_j + t_{j+1} \leq z
\end{cases}
\]

(2.36)

where the left and right overlapping thicknesses $t_j$ and $\tilde{r}_j$ for the $j$th interval are defined as

\[
l_j = y(z_j - d_{j-1}), \quad \tilde{r}_j = y(d_j - \bar{z}_j)
\]

(2.37)

where $z_j$ and $\bar{z}_j$ are the minimum and maximum of the projected data points in the $j$th interval and $y$ is a real parameter between zero and one and controls the overlapping of gate functions for different intervals.

A smooth gate function is defined with two constraints: First, it is equal to one for all the projected data points in its corresponding interval and second, the summation of the smooth gate functions for any point in $R$ is equal to one. The first constraint guarantees that a HARP approximation with smooth characteristic functions $\tilde{X}$, defined using smooth gate functions $\tilde{\Gamma}$, has the same predicted outputs for training data points as a HARP approximation with ordinary characteristic functions $X$. The second constraint ensures that the
predicted output for a point in the overlapping boundary between two subdomains to be a weighted sum of the outputs individually predicted by the subdomain approximations of these subdomains. This property can be mathematically expressed for two adjacent, offspring subdomains as follows

\[ F(x) = \tilde{F}_j(\psi(x; c))\theta_j(x) + \tilde{F}_{j+1}(\psi(x; c))\theta_{j+1}(x) \quad \text{if} \quad d_j - \bar{t}_j \leq \psi(x; c) \leq d_j + t_{j+1} \]  

(2.38)

where the overlapping boundary between the subdomains is \( \psi^{-1}[d_j - \bar{t}_j, d_j + t_{j+1}] \) which by using Eqn. (2.37) can be written as \( \psi^{-1}[(1 - \gamma)d_j + \gamma \bar{t}_j, (1 - \gamma)d_j + \gamma t_{j+1}] \). The overlapping boundary decreases as the value of \( \gamma \) decreases.

Based on the theory of fuzzy subsets, the smooth characteristic function \( \tilde{x} \) represents the membership characteristic function whose membership set is \([0, 1]\) (Kaufmann 1975). During the HARP algorithm, the input domain \( D \) is partitioned into fuzzy subdomains whose membership characteristic functions \( \tilde{x} \) are recursively built according to Eqn. (2.33) by replacing \( \Gamma \) with \( \tilde{\Gamma} \). The partition \( C \), defined in Eqn. (2.2), with fuzzy subdomains take the form

\[ C = \{D_i : D = \bigcup_{i=1}^{s} D_{i, 0.5}, \bigcap_{i=1}^{s} D_{i, 0.5} = \phi \quad \text{if} \quad i \neq j, \quad \sum_{i=1}^{s} \tilde{x}_{D_i}(x) = 1 \ \forall x \in D \} \]  

(2.39)

where \( D_{i, 0.5} \) is the ordinary set of level 0.5 of the fuzzy subdomain \( D_i \). Figure 2.10 shows a typical HARP partition and the ordinary and the membership characteristic functions for one of the subdomains.

We define smooth gate functions such that their overlapping regions do not contain any projected data points, as shown in Fig. 2.9. Therefore based on the recursive equation (2.33), the fuzzy boundaries of subdomains \( D_j \), \( \{x \in D : 0 < \tilde{x}_{D_j} < 1\} \), do not contain any training data points. Consequently, the value of the overlapping parameter \( \gamma \), which controls the Lebesgue measure of fuzzy boundaries, does not affect the approximation error for the training set. The membership characteristic function modifies the approximation in the fuzzy boundary between two neighboring subdomains by weighting the sum of their individual approximations. when \( \gamma \) is set to zero the smooth gate function \( \tilde{\Gamma} \) becomes the step function \( \Gamma \), \( \tilde{x} \) converts to the ordinary characteristic function \( x \), subdomains \( D_j \) become nonfuzzy sets, and mismatching of the subdomain approximations occurs at the boundary of neighboring subdomains. From now on, we refer to subdo-
mains, partitions, and partitioning schemes corresponding to $\gamma=0$ as the nonfuzzy subdomains, partitions, and partitioning schemes, respectively, and those corresponding to $\gamma=1$ as the fuzzy ones.

### 2.4 HARP Algorithm for Multivariate, Mapping Approximation

The proposed HARP algorithm builds a multivariate function to reasonably approximate the given data using a local approximation method. The constructed approximation takes the form

$$ F(x) = \sum_{i=1}^{s} \tilde{\chi}_{D_i}(x; \gamma) \Phi(\theta_i(x; w_i); l_i, u_i, \mu) $$

(2.40)

where $C$ is the final portion of the input domain $D$ constructed by HARP, $s$ is the number of subdomains in $C$, $D_i$ is the $i$th subdomain in $C$, $\theta_i$ is the subdomain approximation corresponding to the $i$th subdomain $D_i$, $w_i$ is the vector of estimated parameters for $\theta_i$, $\Phi$ is the univariate squashing function defined in Eqn. (2.25), and its parameters are $l_i$, $u_i$, and $\mu$ defined in Section 2.2. The membership characteristic function $\tilde{\chi}_{D_i}$ for the $i$th final subdomain is computed by using Eqn. (2.33) as follows

$$ \tilde{\chi}_{D_i}(x; \gamma) = \prod_{j \in S_i} \tilde{r}(\psi(x; c_j); d_j, \gamma) $$

(2.41)

where $\psi$ is the partitioning function, $c_j$ is the vector of random parameters for the $\psi$ function at the $j$th split, $\tilde{r}$ is the smooth gate function defined in Eqn. (2.36) and its overlapping parameter is $\gamma$, $d_j$ is the vector of
splitting thresholds for the $j$th split, and the set of indices $\mathcal{F}_i$ is the list of hierarchical splits generated during the HARP algorithm to determine the $i$th final subdomain $\mathbb{D}_i$. An example partitioning is shown in Fig. 2.6. The set of splits for the $\mathbb{D}_3$ subdomain for example is $\mathcal{F}_3 = \{1, 2, 4, 6\}$ and for $\mathbb{D}_6$ is $\mathcal{F}_6 = \{1, 3, 5\}$.

By combining Eqns. (2.40) and (2.41) the structure of the approximation built by the HARP algorithm takes the following form

$$F_C(x) = \sum_{i=1}^{s} \Phi(\theta_i(x; w_i^r); l_i, u_i, \mu) \prod_{j \in \mathcal{F}_i} \tilde{F}(\psi(x; c_j); d_j, \gamma)$$

(2.42)

The multivariate function $F_C$ belongs to the collection $\mathcal{L}_\mathbb{D}$ if the subdomain approximation $\theta$ and the partitioning function $\psi$ belong to $\mathcal{L}_\mathbb{D}$. The free parameters $\mu$ and $\gamma$ are selected by the experimenter and values $\mu = 0$ and $\gamma = 1$ are recommended. We have explained in Section 2.2 and 2.3 how to select the functions $\theta$ and $\psi$ and to compute their parameters.

When the data are available before the analysis, the proposed HARP method can be summarized in the following algorithm:

**Step 0.** Set the input domain $\mathbb{D}$ to be the parent subdomain. Initialize the set $J$ of poorly approximated subdomains to be empty.

**Step 1.** Fit the selected $\theta$ function to the data in the parent subdomain by minimizing the chosen fitness index through a parameter estimation process.

**Step 2.** Compute the termination criterion for the fitted function $\theta$ for the training data in the parent subdomain. If the criterion is less than the acceptable tolerance then go to **Step 7**.

**Step 3.** Partition the parent subdomain to $r$ offspring subdomains ($r = 2$ is recommended) and add them to the set $J$ of the poorly fit subdomains. For partitioning do the following steps.

**Step 4.** Randomly change the parameters $c$ of the partitioning function such that it is not constant for the data points in the parent subdomain. Store the parameters $c$.

**Step 5.** Select splitting thresholds $\{d_i\}_{i=1}^{r-1}$ to build the partition $C$ in the range of $\psi$. Store the splitting thresholds and maximum and minimum values of $\psi$ for the data points in each offspring subdomain $\{z_i, z_i\}_{i=1}^{r}$ in order to determine function $\tilde{F}$ for the $r$ generated offspring subdomains.

**Step 6.** Assign the training data of the parent subdomain to their corresponding offspring subdomains where the training data set for the $i$th offspring subdomain is the set of training data points $x$ of the parent subdomains whose projected value $\psi(x; c)$ are between the threshold values $d_{i-1}$ and $d_i$. Go to **Step 8**.

**Step 7.** Store the parameters of $\theta$ and the maximum and minimum values of estimated $\theta$ for the data points in the parent subdomain in order to define the squashing function $\Phi$.

**Step 8.** If the set $J$ is empty, then terminate, otherwise pick a poorly approximated subdomain, delete it from the set $J$, set it to be the parent subdomain, and go to **Step 1**.
If the subdomain parameter estimation has an explicit solution for the selected $\theta$ function and fitness index then there is no iterative process in the HARP method and the procedure quickly reduces the error of approximation for the training set to the acceptable level. In the next section two numerical simulations are used to show the capabilities of the HARP method.

2.5 Numerical Simulations

To illustrate the HARP algorithm, we solve two data fitting problems: The XOR problem and a two-dimensional surface fit. For these problems, we choose the subdomain approximation $\theta$ to be a linear function. A term truncation scheme is used to downsize $\theta$ for small subdomains. We use the sum of squared residuals as the fitness index and since $\theta$ is linear in its parameters, we can compute the parameters explicitly by least squares. A ramp squashing function with $\mu=0$ is composed with the subdomain approximation $\theta$. We accept a subdomain approximation when its maximum training residual is less than 0.01. For the subdomain partitioning process, we use the linear partitioning function and set the number of splits for each subdomain $r$ to be two. We select splitting thresholds by the $P^3_\mu$ scheme. The generated subdomains are fuzzy unless we mention otherwise.

The XOR Problem. First the HARP algorithm is used to solve the two dimensional XOR (parity) problem. The input domain is $\{[0, 1] \times [0, 1]\}$ and the training set is $\{(1, 1, 0), (1, 0, 1), (0, 1, 1), (0, 0, 0)\}$. The selected linear $\theta$ function cannot fit the training data. Therefore we randomly partition the input domain into two subdomains. The linear $\theta$ can then fit the data in each subdomain and the total error of approximation for the training set becomes zero. The behavior of the approximations constructed by HARP is shown in Fig. 2.11 for two different partitions of the input domain. Figures 2.11(a) and 2.11(c) present the output of the constructed mapping for the input domain using an ordinary characteristic function and Figs. 2.11(b) and 2.11(d) show the behavior of the constructed mapping with a membership characteristic function. The fuzzy solutions are smoother than the nonfuzzy solutions.

Two-dimensional Surface Fit. The actual function $G_{r_2}$, with domain $\{[0, 2] \times [0, 1]\}$, can be described by

$$G_{r_2}(x) = \begin{cases} 
1 & \text{if } \frac{1}{2} \leq x_2 - x_1 \\
2(x_2 - x_1) & \text{if } 0 \leq x_2 - x_1 \leq \frac{1}{2} \\
\frac{1}{2} \cos \left( 4\pi \left[ \left( x_1 - \frac{3}{2} \right)^2 + (x_2 - \frac{1}{2})^2 \right]^{\frac{1}{2}} \right) + 1 & \text{if } \left( x_1 - \frac{3}{2} \right)^2 + (x_2 - \frac{1}{2})^2 \leq \frac{1}{16} \\
0 & \text{otherwise} 
\end{cases}$$

(2.43)

and is shown in Fig. 2.12(a). A set of 441 data points on a $21 \times 21$ regular grid was used to build the approximation. The constructed approximation was run to predict the output for 3721 test data points on a $61 \times 61$ regular grid. Figure 2.13 presents the evolution of the error on the training set during the HARP training process for one of the random partitions. The maximum and root-mean-squared (RMS) error decreases very rapid-
Fig. 2.11 Solutions for the XOR problem

ly as the number of generated subdomains increases. For the HARP training process, the learning curve of the training error versus the number of subdomains is generally non-increasing.

Figures 2.12(b) to 2.12(f) show the intermediate approximations constructed by HARP for its intermediate partitions. The random partitioning of the input domain and the localized subdomain approximations are clearly indicated in these figures. The approximation constructed by HARP is shown in Fig. 2.12(f) for a random partition of the input domain. The developed approximation has preserved the main features of the actual mapping $G_r$ and has a good generalization. Although the constructed approximation is continuous, it has disturbances along the interior subdomains. However, based on the localized character of HARP, the disturbances in the constructed approximation are local. The effect of the squashing function is evident in Figs. 2.12(b) to 2.12(e) where the subdomain approximations are bounded between zero and one. The developed HARP partition of the input domain is shown in Fig. 2.14. The generated subdomains are convex and have irregular shapes. They are significantly concentrated in the regions where the actual function has complex behavior. Some of the topology of the function is indicated by the shaded features in the figure. As we mentioned, the distribution of subdomains in a HARP partition is influenced by the distribution of training data points and the complexity of the actual mapping. Here, the training data points are uniformly distributed within the input domain. Therefore, the nonuniform distribution of subdomains is caused by the nonhomogeneity
Fig. 2.12 Evolution of a HARP solution for the two-dimensional surface fit problem
of the actual mapping; that is, different relationships hold between input variables in different regions of the input domain.

Because of the random nature of the partitioning, a series of HARP approximations with the same termination criterion for training data can be constructed for a given training set. The difference among the behaviors of these possible solutions decreases as the number of training data points increases. For this example 50 of these solutions were developed. The average of subdomains in the partition of the input domain was 87 and for the test set the average of the maximum error and root-mean-square (RMS) error were 0.156 and 0.0166, respectively.

2.6 A Monte Carlo Strategy for Data-based Mapping Approximation

The HARP partition $C$ is a random partition of the input domain generated by the HARP partitioning process and the HARP approximation $F_C$ is a function of $C$. The collection of possible HARP partitions, $\mathcal{P}_H$, contains an infinite number of partitions for a given training data set and a selected subdomain approximation because of the random nature of the partitioning process. Therefore, by changing the seed of the random number generator, one can generate an infinite number of HARP approximations $F_C$ with minimum error for the training data. Since the constructed mappings $F_C$ are built from the random partitions of the input domain and these partitions are independent of one another, the probability that they are equal to one another is zero.

Fig. 2.13 Learning curves of the HARP algorithm for the two-dimensional surface fit problem

Fig. 2.14 HARP partition for the two-dimensional surface fit problem
Furthermore, the HARP approximations have bounded outputs. Therefore, the predicted output $F_C(x)$ for a point in the input domain is a bounded random variable and, consequently, should have bounded moments including the expected value and standard deviation.

One can use one of the HARP approximations to predict an output for any point in the input domain. However, a more statistically-robust predicted output is the expected value of the population of bounded outputs predicted by the HARP approximations. A statistical estimate of the expected value of random, predicted outputs $F_C(x)$ can be obtained using a sample of the HARP approximations. The sample mean average is an estimate for the expected value. Hence we use a random sequence of $p$ seeds for the random number generator in the HARP algorithm to construct a sample of $p$ equally plausible HARP approximations whose functional representation can be simplified as

$$F_C(x) = \sum_{i=1}^{s_j} \mathcal{Z}_{D_{ij}}(x) \Phi(\theta_{ij}(x)) \quad j = 1, \ldots, p$$

(2.44)

where the index $j$ represents the $j$th partition $C_j$ of the input domain which develops the $j$th HARP approximation, $s_j$ is the number of subdomains in the $j$th partition, $\mathcal{Z}_{D_{ij}}$ is the membership characteristic function for $D_{ij}$, the $i$th subdomain of $C_j$ whose corresponding subdomain approximation is $\theta_{ij}$, and $\Phi$ is the squashing function. A series of statistical indices can be computed for the sample of predicted outputs $\{F_C(x)\}_{i=1}^p$. The two most important statistical indices are the sample mean average $\bar{F}(x)$ and sample standard deviation $\sigma(x)$. The multivariate functions $\bar{F}(x)$ and $\sigma(x)$ represent the expected value and scatter, respectively, of the HARP predicted outputs for any point $x$ in the input domain and are computed as follows

$$\bar{F}(x) = \frac{1}{p} \sum_{j=1}^{p} F_C(x)$$

$$\sigma^2(x) = \frac{1}{p-1} \sum_{j=1}^{p} [F_C(x) - \bar{F}(x)]^2$$

(2.45)

We represent the solution of the data fitting problem as the expected value of the population of HARP approximations and we use a random sequence of numbers to construct a sample of the population, from which a statistical estimate of the expected value can be obtained. This methodology is known as the Monte Carlo method. We apply a Monte Carlo strategy on top of the HARP algorithm and we refer to the proposed procedure as the MC-HARP method.

The MC-HARP method approaches an actual mapping by building a sample of local approximations. The sample of approximations built by the MC-HARP method are plausible; that is, they satisfy the HARP termination criterion for the training set. Furthermore, these approximations are independent of one another, so they can be built and run in parallel. The MC-HARP solution for the data fitting problem is the sample mean average mapping $\bar{F}(x)$. The MC-HARP method computes a deviation measure $\sigma(x)$ for each predicted output $F_C(x)$. The beauty of the $\sigma(x)$ measure is that it can be computed for any point in the input domain without knowing the target output. We will show in our numerical simulations and in Chapter Five that the deviation measure
\( \sigma(x) \) is also a confidence index and can be used to select the best complexity for the MC-HARP approximation when the data are noisy.

The multivariate mappings \( \overline{F}(x) \) and \( \sigma(x) \) belong to the collection \( \mathcal{L}_\mathcal{D} \) if the HARP approximations \( F_C(x) \) belong to \( \mathcal{L}_\mathcal{D} \) because these functions, as defined in Eqn. (2.45), can be represented as the scalar summation and multiplication of a finite collection of \( \mathcal{L}_\mathcal{D} \) functions. Hence, the outputs of MC-HARP are \( \mathcal{L}_\mathcal{D} \) compatible.

The sample of solutions built by MC-HARP is similar to a committee of trained experts, \( F_C \), whose responses for each input pattern, \( F_C(x) \), are equally plausible because they have been trained on the same data set. The response of each expert has equal weight. A simple voting, mean averaging, is used to compute the committee response \( \overline{F}(x) \). The deviation among the experts' responses for an input pattern, \( \sigma(x) \), represents a confidence index for the committee response which means that the smaller \( \sigma(x) \) is, the more confident is \( \overline{F}(x) \). Consequently, greater confidence in the training set results in a smaller \( \sigma(x) \) for each training input pattern. All experts trained by MC-HARP converge to the actual mapping as the amount of data increases. In other words, they are consistent. Furthermore, the difference among the responses of these experts decreases as the amount of training data increases. In the following section we study the characteristics of the MC-HARP method through a numerical simulation for the two-dimensional surface fit defined in Section 2.5.

### 2.7 Numerical Simulation for MC-HARP

The MC-HARP method was used to approximate the two-dimensional function \( G_{r2} \) defined in Eqn. (2.43) and shown in Fig. 2.12(a). The selected subdomain approximation and partitioning process are the same as those chosen for the HARP algorithm in Section 2.5. Three different training sets of data points on \( 9 \times 9, 21 \times 21, \) and \( 41 \times 41 \) regular grids were used to build the approximation. The constructed approximations were used to predict the outputs for test data points on a \( 61 \times 61 \) regular grid. The sample size \( p \) of MC-HARP is set to be 50.

Figure 2.15 shows two HARP random partitions of the input domain developed through the MC-HARP strategy for the data set with \( N=441 \) data points. The generated subdomains have irregular shapes and are significantly concentrated in the regions where the actual mapping has complex variations such as the ramp and
peak areas. The behavior of a typical approximation built by HARP for different training sets is shown in Fig. 2.16(a). The constructed HARP approximations have preserved the main features of the actual function with localized disturbances. Figures 2.16(b) and 2.16(c) present the behavior of the MC-HARP approximation \( \bar{F}(x) \) and its corresponding deviation function \( \sigma(x) \) constructed for different training sets. It is evident that by increasing the amount of training data, the MC-HARP approximation \( \bar{F}(x) \) converges to the actual mapping faster and with less localized disturbances that each individual HARP approximation \( F_c(x) \). The standard deviation \( \sigma(x) \) is very small for the training data points. Also, the deviation measure \( \sigma(x) \) for MC-HARP predicted outputs is smaller and decreases more rapidly with the number of data points for regions in the input domain where the complexity of the actual mapping is low like the ramp region than more complex peak region. In other words, the MC-HARP deviation measure \( \sigma(x) \) follows the complexity of the data.

2.8 Conclusions

In this chapter, a robust method for approximating multivariate mappings, based on the concept of hierarchical adaptive random partitioning (HARP), has been presented. The basic nature of the method is local approximation. The input domain is partitioned into subdomains and independent local approximations are built for each subdomain. The concepts of inverse image, partitioning function, and characteristic function are used to represent a HARP approximation by simple operations of summation, multiplication, and composition of univariate functions. Subdomain training processes are independent of one another, so these computations can be done in parallel. A possibly large number of small, independent systems of equations are solved in the HARP training process, in contrast with global, parametric methods which require the solution of fairly large system of equations.

The HARP partitioning has the flexibility to adapt to the behavior of the data and is approximation-error-driven. More subdomains are generated where the data have complex behavior. The HARP approximation shows good generalization because it captures the essential features of the data and its disturbances are localized. The subdomains are randomly divided, therefore the partitioning is fast and computationally efficient, and well suited to parallel processing. The HARP method gives rise to many equally plausible solutions to a data fitting problem, in contrast to other methods which have difficulty finding even one solution. The HARP algorithm does not require user interaction. Every operation in the HARP method is logical and has a sound mathematical basis. Thus, the mathematical anatomy of the HARP approximation is transparent.

Fuzzy partitioning is added to the HARP algorithm to improve its generalization. Fuzziness enforces continuity of the mapping constructed by the HARP and smooths the mismatching of the local approximations in the neighboring subdomains. Furthermore, we can apply a Monte Carlo strategy on top of the HARP algorithm and develop a new method for data-based mapping approximation called the MC-HARP method. The MC-HARP approximation is the mean average of a sample of HARP approximations. For each input pattern, the MC-HARP method can compute a confidence index for the predicted output.

By increasing the amount of data, superposition of HARP approximations through the Monte Carlo strategy makes the MC-HARP approximation converge to the actual mapping more uniformly and with less localized disturbances than each individual HARP approximation. Therefore, the proposed Monte Carlo strategy
Fig. 2.16 Behavior of the MC-HARP method

(a) Behavior of a typical HARP approximation $F_c(x)$ for different training sets

(b) Behavior of the Me-HARP approximation $F(x)$ for different training sets

(c) MC-HARP deviation function $\sigma(x)$ for different training sets
improves the generalization of the HARP approximation for a fixed amount of training data. The constructed MC-HARP approximation shows good generalization because it has the flexibility of the local approximations to adapt to complex, nonhomogeneous functional behavior and the smoothness of the global approximations to capture the global features of the data. The HARP algorithm with fuzzy partition gives local adaptivity and continuity to the MC-HARP method and the Monte Carlo strategy controls its global smoothness. The MC-HARP method preserves all the main characteristics of the HARP algorithm, namely structural self-organization, fast learning, and automatic processing. The independence of subdomain approximations and of HARP approximations make the MC-HARP method highly parallelizable.
CHAPTER THREE
Numerical Simulation Studies of MC-HARP and Neural Network Representation of MC-HARP

“Order and simplification are the first steps toward the mastery of a subject; the actual enemy is the unknown.”

Thomas Mann

“I have hardly ever known a mathematician who was capable of reasoning.”

Plato

In previous chapter, we presented the MC-HARP method for data-based approximation of multivariate mappings, based on the concepts of hierarchical adaptive random partitioning (HARP), fuzziness, and Monte Carlo approximation. In this chapter, we study the performance of the MC-HARP method through numerical simulations. We then show the relationship between neural networks and MC-HARP. We study the behavior of the MC-HARP approximation with respect to the number of HARP partitions, amount of training data, complexity of the subdomain approximation, partitioning scheme, termination tolerance, and dimension of the input domain. Further, we show how the MC-HARP method can be used to simultaneously train and build mapping neural networks. The training process of MC-HARP is compatible with the training process of a mapping neural network. An MC-HARP approximation can be modeled as a modular, feedforward neural network with two hidden layers. The basic module of the MC-HARP network is a neural network with one hidden layer built by the HARP algorithm.

3.1 Performance and Complexity Indices

In this section we introduce a few appropriate indices for use in probing the behavior of the MC-HARP method by simulation. To measure the complexity of an MC-HARP approximation $F$, we use the number of parameters and subdomains it has as two complexity indices. The number of parameters $K$ in the approximation $F$ is defined as follows

$$K = \frac{1}{p} \sum_{i=1}^{p} K_{c_i}$$

(3.1)

where $p$ is the MC-HARP sample size and $K_{c_i}$ is the number of parameters in the HARP approximation $F_{c_i}$, equal to the sum of numbers of parameters in all subdomain approximations used to build $F_{c_i}$. The complexity index $\bar{K}$ is the sample mean average of the numbers of parameters in the $p$ approximations built by HARP.
Similarly, we define the number of subdomains for the MC-HARP approximation $\bar{F}$ to be the sample mean average of the numbers of subdomains in HARP partitions built during the training process of MC-HARP and is computed as

$$\bar{s} = \frac{1}{p} \sum_{i=1}^{p} s_{C_i}$$

where $\bar{s}$ is the number of subdomains for $\bar{F}$ and $s_{C_i}$ is the number of subdomains in the $i$th HARP partition $C_i$ corresponding to the HARP approximation $F_{C_i}$. For an MC-HARP approximation, $K$ is generally a nondecreasing function of $\bar{s}$ meaning that, by increasing the number of subdomains in $\bar{F}$, its number of parameters increases. Furthermore, $K$ is a complexity measure. The smaller $K$ is, the simpler an MC-HARP approximation is.

To study the performance of the MC-HARP approximation $\bar{F}$, three indices are considered for the test set: the average standard deviation $\sigma_F$, the root mean squared error $RMS_F$, and the maximum error $e_{max}$. The performance index $\sigma_F$ represents the mean average of the deviation of HARP approximations $F_{C_i}$ about their mean $\bar{F}$ computed for the test set and is defined as

$$\sigma_F = \frac{1}{N_t} \sum_{x \in \mathcal{X}_t} \sigma(x)$$

where $\sigma(x)$ is the standard deviation computed by MC-HARP, as defined in Eqn. (2.45), for a point $x$ in the set of test data points $\mathcal{X}_t$ and $N_t$ is the number of test data points. The deviation index $\sigma_F$ is a measure of precision of an MC-HARP approximation. The smaller standard deviation is, the more precise an MC-HARP approximation is.

The root squared error $RMS_F$ represents the distance between the sample mean average $\bar{F}$ and the actual mapping $G$ and is defined as

$$RMS_F = \left[ \frac{1}{N_t} \sum_{x \in \mathcal{X}_t} (\bar{F}(x) - G(x))^2 \right]^{1/2}$$

The performance index $RMS_F$ is a measure of accuracy for an MC-HARP approximation $\bar{F}$. The smaller the prediction error $RMS_F$ is, the more accurate $\bar{F}$ is. Another performance measure is the maximum error $e_{max}$ that is defined as

$$e_{max} = \max \{|\bar{F}(x) - G(x)| : x \in \mathcal{X}_t\}$$

The performance indices $RMS_F$ and $e_{max}$ indicate the prediction error of an MC-HARP approximation $\bar{F}$ and are measures for the generalization of $\bar{F}$. The smaller these indices are, the better the generalization of $\bar{F}$ is.

The defined performance indices represent the approximation error when the data are not noisy. Another source of error is the noise in data. In the next chapter we study the performance of MC-HARP for noisy data. In our numerical simulations in this chapter, the training data are noise-free.
3.2 Numerical Simulations

For all numerical simulations in this section, we choose the subdomain approximation $\theta$ to be a polynomial. A term truncation scheme is used to downsize $\theta$ for small subdomains. We use the sum of squared residuals as the fitness index and since $\theta$ is linear with respect to its parameters, we compute its parameters by least squares. A ramp squashing function with $\mu=0$ is composed on the subdomain approximation $\theta$. We use the maximum training residual as the termination criterion. We use the linear partitioning function to randomly partition subdomains and set the number of splits for each subdomain, $r$, to be two.

3.2.1 Effect of Sample Size

We use the two-dimensional $G_{r^2}$ function, defined in Eqn. (2.43), to generate training and test data. The tolerance value $\epsilon$ for the termination criterion is set to be 0.01. The $\mathcal{P}_H$ scheme is used to select splitting thresholds. The generated subdomains are fuzzy, meaning that the overlapping parameter $r$ for the characteristic function is set to be one. Nine different training sets of data points on $9 \times 9$, $13 \times 13$, $17 \times 17$, $21 \times 21$, $25 \times 25$, $29 \times 29$, $33 \times 33$, $37 \times 37$, and $41 \times 41$ regular grids are used to build the approximation. The constructed MC-HARP approximations are used to predict the outputs for test data points on a $61 \times 61$ regular grid. The sample size $p$ for the MC-HARP method is selected to be between one and 30.

Figure 3.1 shows the variation of performance measures, $RMS_F$ and $\sigma_F$, and complexity measures, $\tilde{r}$ and $\tilde{K}$, with respect to the sample size $p$ for different training sets. It is evident that the performance and complexity indices become steady when the sample size is large enough. This characteristic is typical for a Monte Carlo method and can be used to select an adequate sample size. The computational intensity of MC-HARP is directly related to the sample size $p$. Hence, the selected value should not be too big. On the other, the MC-HARP approximation built using a small sample of HARP approximations has less accuracy than the MC-HARP approximation with large $p$ value, as shown in Fig. 3.1. Therefore, a compromise between accuracy and computational effort is essential. A sample is sufficiently large when the performance and complexity measures do not significantly change with additional HARP partitions. One can observe in Fig. 3.1 that a sample size of about ten is adequate for our numerical simulation.

The observed variation of performance measures, $RMS_F$ and $\sigma_F$, with respect to the sample size $p$ can be modeled using the statistical sampling theory. The output $\bar{F}(x)$ predicted by the MC-HARP approximation $\bar{F}$ for a point $x$ in the input domain, as defined in Eqn. (2.45), is the mean average of a sample of bounded, random outputs $\{F_C(x)\}_{i=1}^p$ predicted by HARP approximations. Consequently $\bar{F}(x)$ is also a bounded, random variable. Let $\mu_F(x)$ be the expected value of the random variable $F_C(x); E_C[F_C(x)] = \mu_F(x)$. Since the performance measure $RMS_F$, defined in Eqn. (3.4), is a function of $F_C(x)$ it is a random variable. Knowing that $E_C[F_C(x) - \mu_F(x)] = 0$, the expected value of $RMS_F$ can be computed as follows

$$E_C[RMS_F^2] = \frac{1}{N_x} \sum_{x \in \mathbb{X}_x} \left[ E_C[(\bar{F}(x) - \mu_F(x))^2] + E_C[(\mu_F(x) - G(x))^2] \right]$$

(3.6)

assuming that the approximation errors $(\bar{F}(x) - G(x))$ for test data points are independent of one another. The second term $(\mu_F(x) - G(x))^2$ on the right hand side of Eqn. (3.6) is not a random variable and represents the
difference between the actual output $G(x)$ and the expected output computed by the random HARP approximations. Since the random variables $F_C(x)$ are bounded, independent, and identically distributed, they and their mean $\bar{F}(x)$ have bounded moments. By defining the second central moment, or variance, $\text{Var}_F(x)$ as

$$\text{Var}_F(x) = E_c[(F_C(x) - \mu_F(x))^2]$$  \hspace{1cm} (3.7)

the second central moment for the mean value $\bar{F}(x)$ is computed as follows

$$E_c[(\bar{F}(x) - \mu_F(x))^2] = \frac{\text{Var}_F(x)}{p}$$  \hspace{1cm} (3.8)

Now, substituting Eqn. (3.8) into Eqn. (3.6) leads to

$$E_c[RMS_F^2] = \frac{1}{pN_c} \sum_{x \in B} \text{Var}_F(x) + \frac{1}{N_t} \sum_{x \in B_t} (\mu_F(x) - G(x))^2$$  \hspace{1cm} (3.9)

Equation (3.9) indicates that the expected value of the performance measure $RMS_F^2$ converges to a fixed value, shown by the second term on the right hand side of Eqn. (3.9), as the sample size increases and the rate of convergence is $1/p$. In other words, by increasing the sample size $p$, the approximation error $RMS_F^2$ converges to a bias value between the expected approximation $\mu_F(x)$ of HARP approximations and the actual

---

**Fig. 3.1 Variation of performance and complexity indices with respect to the sample size for different numbers of data**

48
mapping $G(x)$. The bias value is a function of the amount of training data and decreases as the amount of data increases. Figure 3.1 shows that the performance measure $RMS_F$ decays as the sample size increases and its limit value, which is greater than zero, decreases as the amount of data increases.

The deviation measure $\sigma(x)$ is also a function of the random variable $F_c(x)$ and consequently is a bounded, random variable. The expected value of the sample variance $\sigma(x)^2$ is $Var_F(x)$. The 100$(1-\alpha)$ percentile confidence interval on $\sigma(x)^2$ is

$$\frac{\chi^2_\alpha Var_F(x)}{p-1} \leq \sigma^2(x) \leq \frac{\chi^2_{(1-\alpha/2)} Var_F(x)}{p-1}$$

(3.10)

where $\chi^2_\beta$ is the $100\beta$ percentile point of the $\chi^2$ distribution with $(p-1)$ degrees of freedom. For example, for a sample size $p$ equal to 10, the 95 percentile point $\chi^2_{0.95}$ is equal to 16.9. The confidence interval in Eqn. (3.10) represents an interval containing the random variable $\sigma(x)^2$ with the probability of $(1-\alpha)$. By increasing the sample size $p$, the number of degrees of freedom for the $\chi^2$ distribution increases and $\chi^2$ converges to a normal distribution with mean and variance equal to $p-1$. Consequently, the percentiles $\chi^2_{(1-\alpha/2)}$ and $\chi^2_{\alpha/2}$ converge to $p-1$ and, based on Eqn. (3.10), the sample variance $\sigma(x)^2$ converges to its expected value $Var_F(x)$. Hence, the performance measure $\sigma_F$, that is the mean average of the deviation measures $\sigma(x)$ for the test data points, is convergent with respect to the sample size $p$. This characteristic can be observed in Fig. 3.1 where the performance index $\sigma_F$ becomes steady when the sample size $p$ is large enough. The limit value of $\sigma_F$ for large $p$ is a function of the amount of training data and decreases as the amount of data increases, as shown in Fig. 3.1.

The accuracy and precision of an MC-HARP approximation, measured by the performance indices $RMS_F$ and $\sigma_F$, increases as the amount of training data increases, indicating that the MC-HARP approximation is consistent. The complexity indices $\bar{s}$ and $\bar{K}$ become steady when the sample size $p$ is large enough. The limit values of these complexity indices with respect to the sample size $p$ increase as the amount of training data increases. However the rates at which the numbers of parameters and subdomains increase are slower than the rate at which the number of data increases. Actually the complexity measures $\bar{s}$ and $\bar{K}$ have limit values when the amount of data is large enough for the selected termination tolerance. We study this characteristic in the next numerical simulation.

For a fixed amount of data, there is a critical sample size above which, for the MC-HARP approximations with samples larger than this critical size, the complexity or performance indices do not change significantly. This critical sample size is a reasonable candidate for the MC-HARP sample size. It is evident in Fig. 3.1, that the critical sample size for the performance indices, $RMS_F$ and $\sigma_F$, decreases as the amount of data increases. However for the complexity indices, $\bar{s}$ and $\bar{K}$, the critical sample size increases as the amount of data increases. A reliable measure for selecting the sample size $p$ is the performance index $RMS_F$. The problem with this measure is that often we do not have enough data to hold out for a test set. Based on our numerical simulations, we recommend using the performance measure $\sigma_F$ for selecting the sample size. because the measure $\sigma_F$ can be always computed for a set of test data points without knowing
their corresponding target output. Also the index $\sigma_F$ is a performance measure and using $\sigma_F$ for selecting $p$ leads to smaller sample size for MC-HARP than using complexity indices.

3.2.2 Effect of the Complexity of the Subdomain Approximation

By increasing the complexity of the subdomain approximation $\theta$, the number of subdomains for an MC-HARP approximation might be reduced. However the subdomain training becomes more complicated, which might slow down the training process, the complexity of the constructed MC-HARP approximation might increase, and, based on the behavior of the actual mapping, we might not gain any significant improvement in the generalization.

In this numerical example, we use two-dimensional polynomials of first degree through fourth degree for the subdomain approximation to fit the actual mapping $G_{r_2}$. The tolerance value $\epsilon$ for the termination criterion is set to be 0.01. The fuzzy $\mathcal{F}_P$ scheme is used to select splitting thresholds. The selected sample size $p$ is 30. The nine training data sets, used in the previous example, with the number of data from 81 to 1681 are used to build the approximation. The test data are selected on a $61 \times 61$ regular grid.

Figure 3.2 shows the variation of performance and complexity measures with respect to the amount of training data for different subdomain approximations. As we observed in the previous numerical example, the prediction error $RMS_F$ and the deviation measure $\sigma_F$ decrease as the number of data increases. However increasing the complexity of the subdomain approximation does not improve the performance of the constructed MC-HARP approximations. For a given amount of data, the performance measures $RMS_F$ and $\sigma_F$ are almost the same for MC-HARP approximations with a linear $\theta$ through quartic $\theta$. One reason for this behavior is the simplicity of the actual mapping for the majority area of the input domain except the peak region. A polynomial of higher degree cannot efficiently use its complexity to fit the simple ramp type of behavior in $G_{r_2}$. In other words, the higher degree subdomain approximations should set the parameters corresponding to their higher terms to zero in order to fit a ramp behavior. The increasing of complexity without a significant improvement in performance is typical for nonhomogeneous actual mappings. In order to overcome this problem, we recommend always to use simple subdomain approximations unless a priori knowledge about the mapping approximation problem dictates otherwise.

As we can observe in Fig. 3.2, by increasing the amount of training data, the numbers of parameters and subdomains in the MC-HARP approximation increase. However the rates of increase of $\bar{s}$ and $\bar{K}$ decrease as the number of data increases. Actually the rates of increase of $\bar{s}$ and $\bar{K}$ are slower than the rate of increase for the number of data $N$. In other words, the ratios $\bar{s}/N$ and $\bar{K}/N$ converge to zero as $N$ increases. This characteristic of MC-HARP is shown in Fig. 3.2 by the saturation of curves $\bar{s}(N)$ and $\bar{K}(N)$ for large amounts of data points. The slower rate of increase for $\bar{s}$ in comparison to the rate of increase of data makes the average number of data points in each subdomain, represented by $N/\bar{s}$, increase as the number of data increases. This trend indicates that the size of subdomains does not change in the regions of the input domain that the assumed subdomain approximation $\theta$ can fit the actual mapping within the selected tolerance $\epsilon$ for the termination criterion. Hence, the saturation of complexity curves $\bar{s}(N)$ and $\bar{K}(N)$ for large data sets is a general characteristic of MC-HARP. For a selected tolerance value $\epsilon$, a locally-convergent subdomain approximation can fit
data when its corresponding subdomain is small enough. In a region of the input domain, the size of a subdomain required for an acceptable fit by $\theta$ depends on the complexity of the actual mapping in that region and the selected tolerance value, but it is a finite value. Therefore, the maximum number of subdomains, which is directly related to the required sizes of subdomains in different regions of the input domain, is finite. Consequently the complexity measure $\mathcal{S}$ has an upper bound and should saturate with respect to $N$ when the data set is large enough. Furthermore, the number of parameters $K$, which is the mean average of the sum of parameters in subdomain approximations, is directly related to the number of subdomains $\mathcal{S}$ and, like $\mathcal{S}$, should saturate with respect to $N$.

It is evident in Fig. 3.2 that, for a given amount of data, by increasing the complexity of the selected subdomain approximation $\theta$, the number of subdomains decreases and the number of parameters increases. Generally, in MC-HARP, the complexity increase of $\theta$ reduces the number of subdomains. When the number of parameters in $\theta$ is equal to the number of data, the selected $\theta$ can fit the data perfectly (if there is no collinearity) and the number of subdomains is equal to one. Hence, by increasing the complexity of $\theta$, the number of subdomains $\mathcal{S}$ converges to one. The relationship between the number of parameters $K$ and the complexity of $\theta$ depends on the similarity between the behaviors of the actual function and the selected $\theta$ within the input domain. In this numerical example, the higher degree polynomial $\theta$ cannot efficiently use its complexity to

Fig. 3.2 Variation of performance and complexity indices with respect to the number of data for different subdomain approximations
fit the simple nonhomogeneous $G_{12}$ mapping. In other words, there are subdomains in which the corresponding estimated $\theta$ functions have many zero parameters and thus fit data that easily can be represented by a simpler $\theta$ function. Hence an MC-HARP approximation with a complex $\theta$ uses more parameters than an MC-HARP approximation with a simple $\theta$ while they both have the same performances. One should expect that, for an actual mapping that has complex behavior in a large region of the input domain, using more complex $\theta$ functions reduces the number of parameters $K$ because a complex $\theta$ can fit more efficiently the complex local behavior of the actual function than a simple $\theta$. These characteristics of MC-HARP lead us to the recommendation that always use simple subdomain approximations $\theta$ except when the complexity increase of $\theta$ improves the performance or reduces the complexity of the constructed MC-HARP approximation.

3.2.3 Effect of Termination Tolerance

The number of subdomains in each HARP partition $C$ generated during the training process of MC-HARP is a function of the tolerance value $\varepsilon$ for the termination criterion. Consequently, the mean average of the numbers of subdomains in the partitions, $\bar{s}$, is also a function of $\varepsilon$. When $\varepsilon$ is large enough $\bar{s}$ is equal to one, meaning that the selected subdomain approximation $\theta$ can approximate the actual mapping with accuracy $\varepsilon$ and thus there is no need to partition the input domain. By setting $\varepsilon$ equal to zero, $\bar{s}$ reaches its maximum value because we want the HARP approximations $F_C$ to fit the data perfectly. A parametric subdomain approximation $\theta$ can fit data perfectly when its corresponding subdomain is small enough. Hence, by decreasing the tolerance value $\varepsilon$, the smallest subdomain in a HARP partition $C$ shrinks and the number of subdomains in $C$ increases. Indicating that $\bar{s}(\varepsilon)$ is a nonincreasing function of $\varepsilon$ whose maximum and minimum value are $\bar{s}(0)$ and one, respectively.

The number of parameters $K_C$ in a HARP approximation $F_C$ is the sum of numbers of parameters in all subdomain approximations used to build $F_C$. Hence $K_C$ is directly related to the number of subdomains in $C$. Similarly, the number of parameters $\bar{K}$ in an MC-HARP approximation $\bar{F}$ is a nondecreasing function of its number of subdomains $\bar{s}$. Since $\bar{s}(\varepsilon)$ is a nonincreasing function of $\varepsilon$ and $\bar{K}(\bar{s})$ is a nondecreasing function, the complexity measure $\bar{K}$ is also a nonincreasing function of $\varepsilon$. We refer to a curve showing the variation of a complexity index with respect to the tolerance value $\varepsilon$ as the complexity curve. The tolerance value $\varepsilon$ is an indicator of the complexity of the HARP approximations $F_C$ and the MC-HARP approximation $\bar{F}$. At $\varepsilon=0$, $F_C$ perfectly fits the data with the maximum number of subdomains and parameters. When $\varepsilon$ is a large value, there is only one subdomain in $C$, the input domain, and $F_C$ is represented by the subdomain approximation $\theta$. Therefore by increasing the value of $\varepsilon$ from zero to a large value, the HARP approximation $F_C$ moves from a local interpolation to a parametric approximation $\theta$. The same characteristic are true for the MC-HARP approximation $\bar{F}$ as the sample mean average of HARP approximations $F_C$.

In the HARP algorithm, the distribution of subdomains is related to the distribution of data points and the complexity of the actual mapping. For a given data set, the HARP algorithm controls the size of subdomains in any region of the input domain based on the relative complexity of the actual mapping with respect to the selected $\theta$ in that region in comparison with the selected tolerance. Because the HARP partitioning is approximation-error-driven, for a selected tolerance value, HARP attempts to effectively distribute subdo-
mains in the input domain to fit the data within the selected tolerance. Consequently, we can use the tolerance value \( \varepsilon \) to control the distribution, size, and number of subdomains in a HARP partition. Furthermore, the number of parameters in a HARP approximation can also be controlled by \( \varepsilon \). In other words, the tolerance for the approximation error can control an approximation-error-driven partitioning scheme and its corresponding local approximation. Therefore, for a given data set, we can control the complexity of an MC-HARP approximation \( \bar{F} \) by the tolerance value \( \varepsilon \) and the structural details of \( \bar{F} \), including the distribution, size, and number of subdomains, are automatically determined by the HARP algorithm. A HARP partition \( C \), HARP approximation \( F_C \), and MC-HARP approximation \( \bar{F} \) are functions of \( \varepsilon \); i.e., \( C(\varepsilon) \), \( F_C(\varepsilon) \), and \( \bar{F}(\varepsilon) \).

In this numerical example we study how the tolerance value \( \varepsilon \) controls the performance of MC-HARP. We use the \( G_{r2} \) function to generate a training set of data points on a \( 21 \times 21 \) regular grid. The test set, the same as the previous examples, contains data points on a \( 61 \times 61 \) regular grid. The subdomain approximation \( \theta \) is chosen to be a linear function. The sample size \( p \) is set to be 30. The fuzzy \( \mathcal{P}_H \) scheme is used to select splitting thresholds. A series of MC-HARP approximations is built for different tolerance values.

Figure 3.3(a) shows the variation of the complexity indices with respect to the tolerance value \( \varepsilon \). It is evident that the complexity curves \( s(\varepsilon) \) and \( K(\varepsilon) \) are nonincreasing. The MC-HARP approximation \( \bar{F}(0) \) has the largest numbers of parameters and subdomains in the family of MC-HARP approximations \( \bar{F}(\varepsilon) \). The constructed mapping \( \bar{F}(0) \) is the most complex mapping built by MC-HARP for the given data set. Although \( \bar{F}(0) \) is an interpolation, its number of parameters \( K(0) \) is smaller than the number of training data because the simple linear \( \theta \) can represent the actual function \( G_{r2} \) for a large region of the input domain. We define \( \varepsilon_{\text{max}} \) to be the lower bound of tolerance values whose corresponding MC-HARP approximations have only one subdomain; i.e., \( s(\varepsilon_{\text{max}})=1 \). For this numerical example, \( \varepsilon_{\text{max}} \) is equal to one, as shown in Fig. 3.3(a). The number of parameters \( K(\varepsilon_{\text{max}}) \) is equal to the number of parameters in \( \theta \) which for a two dimensional linear \( \theta \) is equal to 3, as shown in Fig. 3.3(a). For the tolerance values greater than or equal to \( \varepsilon_{\text{max}} \), all the HARP approximations \( F_C \) and their mean \( \bar{F} \) are equal to \( \theta \) and their numbers of subdomains and parameters do not change. Hence, none of the complexity indices, \( K \) and \( s \), changes for \( \varepsilon \geq \varepsilon_{\text{max}} \).
To study the performance of MC-HARP, we use the performance indices $a_F$ and $RMS_F$ defined for the test set and two similar indices $a_{F}^{\text{Train}}$ and $RMS_{F}^{\text{Train}}$ for the training set. The indices $a_{F}^{\text{Train}}$ and $RMS_{F}^{\text{Train}}$ are respectively computed using Eqsns. (3.3) and (3.4) with $\mathcal{S}$ and $N$ instead of $\mathcal{S}_t$ and $N_t$. Since these performance indices are function of HARP and MC-HARP approximations, they are all functions of the tolerance value $\varepsilon$. We refer to a curve showing the variation of a performance index with respect to $\varepsilon$ as the performance curve. Figure 3.3(b) shows performance curves for MC-HARP approximations constructed to approximate $G_{\gamma^2}$. Since the performance indices do not change for the tolerance values greater than $\varepsilon_{\text{max}}$, we refer to the interval $[0, \varepsilon_{\text{max}}]$ as the domain of the performance curves. In this section we study the general behavior of the performance curves. In next chapters, we shall study these curves in more details and show how they can be used to deal with noisy data, select the best complexity, and check the adequacy of data.

Both $RMS_F$ and $RMS_{F}^{\text{Train}}$ performance curves are nondecreasing and reach their maximum value when $\varepsilon$ is a large value, $\varepsilon_{\text{max}}$. These curves separate from each other when the tolerance value is small. At $\varepsilon=0$, the performance index $RMS_{F}^{\text{Train}}$ is equal to zero because we perfectly fit the training data. However the $RMS_F$ index is greater than zero for $\varepsilon=0$ indicating the bias between the constructed MC-HARP approximation $F(0)$ and the actual mapping. Since the set of test data points cover the input domain with high resolution, the performance index $RMS_F$ is a reasonable measure for the approximation accuracy. It reaches its minimum when the tolerance value $\varepsilon$ is equal to zero. Therefore, for the given data set, the most accurate MC-HARP approximation is $F(0)$ and consequently the best tolerance value is zero. We will show in next chapters that for noise-free data, the best tolerance value is zero and $F(0)$ has the best complexity. By decreasing the tolerance value $\varepsilon$, the complexity of the constructed MC-HARP approximation increases and its accuracy improves. The increasing rate of accuracy is small when the tolerance value is close to zero. As shown in Fig. 3.3(b) for the tolerance values less than 0.2, the performance index $RMS_F$ does not change significantly. On the other hand, the numbers of subdomains and parameters significantly decrease as $\varepsilon$ increases and reaches the value 0.2. This observation indicates that $F(0.2)$ is simpler than $F(0)$ and has almost the same accuracy. Hence one can always build MC-HARP approximations that are simpler than the best approximation $F(0)$ and have almost the same accuracy by choosing tolerance values close to zero. We will discuss in next chapters how to decrease the complexity without decreasing accuracy.

The deviation curve $\sigma_{F}^{\text{Train}}$ is equal to zero at the end points of its domain, as shown in Fig. 3.3(b). At $\varepsilon=\varepsilon_{\text{max}}$, the deviation measure $\sigma_{F}^{\text{Train}}$ is equal to zero because all the HARP approximations are equal to $\theta$ and there is no deviation among them. At $\varepsilon=0$, the outputs predicted by HARP approximations for each training data point are all equal to the target output for that point. Hence, there is no deviation among the outputs predicted by HARP approximations for each training data point and consequently, the deviation measure $\sigma_{F}^{\text{Train}}$ is equal to zero when $\varepsilon=0$. Since $\sigma_{F}^{\text{Train}}$ is a nonconstant, positive function with respect to $\varepsilon$, and is equal to zero at the boundaries of the interval $[0, \varepsilon_{\text{max}}]$, it should have at least one maximum point in the interval $[0, \varepsilon_{\text{max}}]$, as shown in Fig. 3.3(b). The $\sigma_F$ performance index is equal to zero like $\sigma_{F}^{\text{Train}}$ when the tolerance value is equal to $\varepsilon_{\text{max}}$. However, unlike $\sigma_{F}^{\text{Train}}$, the deviation measure $\sigma_F$ is not equal to zero at $\varepsilon=0$ indicating that the HARP approximations have different values at test data points which are remote from the
training data points. The performance curve $\sigma_F$ is close to the $\sigma_F^{\text{Train}}$ curve for a significant portion of its domain. These curves separate from each other for tolerance values close to zero.

The MC-HARP method can provide two performance measures $\sigma_F$ and $\sigma_F^{\text{Train}}$ for its constructed mappings besides the common performance measures like $\text{RMS}_F$. The deviation curves $\sigma_F$ and $\sigma_F^{\text{Train}}$ present a rich insight into MC-HARP. In next chapters we will show how these deviation indices lead to a new philosophy for measuring the performance of data-based, approximate mapping and introduce a new criterion for selecting the complexity of MC-HARP approximations. Furthermore, the deviation indices, $\sigma_F$ and $\sigma_F^{\text{Train}}$, measures the approximation confidence and can provide reasonable bounds for the approximation accuracy. Also, we will use these deviation indices to develop a new framework for classifying data-based mapping approximation problems based on the quality and quantity of data.

3.2.4 Effect of Fuzziness and Schemes for Selecting Splitting Thresholds

In Section 2.3.2, we introduced three schemes, $\mathcal{P}_k$, $\mathcal{P}_h$, and $\mathcal{P}_i$, for selecting splitting thresholds for the subdomain partitioning process of the HARP algorithm. Furthermore, we developed fuzzy boundaries around HARP subdomains to smooth the mismatching of the local approximations in the neighboring subdomains. The location of fuzzy boundaries in a HARP partition $\mathcal{C}$ is controlled by the splitting thresholds. The overlapping parameter $\gamma$, defined in Section 2.3.4, controls the size of fuzzy subdomains. Besides the random parameters in the partitioning function $\psi$ that controls the alignment of subdomain boundaries, the splitting thresholds and overlapping parameter determine the configuration of a HARP partition $\mathcal{C}$. In this section, we use numerical simulations to study how fuzziness and schemes for selecting splitting thresholds influence the performance of MC-HARP.

We use the two-dimensional XOR problem, defined in Section 2.5, and the $G_{r2}$ function to generate training and test data. For the XOR problem, we choose the subdomain approximation $\theta$ to be a constant function and for the $G_{r2}$ function, we choose $\theta$ to be a linear function. The tolerance value $\varepsilon$ for the termination criterion is set to be 0.01. A set of data points on a $61 \times 61$ regular grid is used to investigate the performance of the constructed MC-HARP approximations. The sample size $p$ is set to be 50 for the $G_{r2}$ surface fitting problem and to be 2000 for the XOR problem.

Figure 3.4 shows the variation of performance measures, $e_{\text{max}}$ and $\sigma_F$, for MC-HARP approximations of $G_{r2}$ with fuzzy and nonfuzzy partitions with respect to the amount of training data. In this example, we use the $\mathcal{P}_h$ scheme for selecting splitting thresholds. As we had also observed in Fig. 3.2, by increasing the amount of data, the approximation error, represented by $e_{\text{max}}$, and the approximation deviation, represented by $\sigma_F$, both decrease for both fuzzy and nonfuzzy approximations. For small data sets, the fuzzy approximation has slightly lower error and deviation than the nonfuzzy approximation. For large data sets, it is evident that fuzziness does not significantly improve the performance. The reason for this behavior can be explained as follows: by increasing the amount of data, the data points become closer to one another. Since the size of a fuzzy boundary is directly related to the values of projected data points $\psi(x)$, the size of the fuzzy boundary decreases as the amount of data increases. Consequently the effect of fuzziness on the performance of the MC-HARP approximation decreases as the amount of data increases.
Figure 3.5 shows the performance curves for MC-HARP approximations with fuzzy and nonfuzzy partitions built for a data set of 441 training data points. It is evident that, for the given data set, fuzziness does not change the accuracy of constructed MC-HARP approximations but slightly decreases their approximation deviation when the tolerance value is small. As we previously mentioned by decreasing the tolerance value \( \varepsilon \), the number of subdomains in HARP partitions increases. Consequently the measure of boundary regions among subdomains increases. Indicating that the influence of fuzziness should be more evident for small tolerance values than the large values. This trend can be observed in the behavior of the deviation curve \( \sigma_F \) in Fig. 3.5. The reason that the \( \text{RMS}_F \) measure is the same for fuzzy and nonfuzzy approximations even for small tolerance values, can be explained as follows: The actual mapping \( G_{r2} \) is simple for the majority of the input domain and for the region with simple data behavior, the MC-HARP superposition of random local approximations develops smoothness similar to the fuzzy smoothness. Therefore, the bias between the constructed mappings and \( G_{r2} \) is not affected by the boundary fuzziness.

Based on trends in Figs. 3.4 and 3.5, one can conclude that fuzziness improves the performance for regions of the input domain where the actual mapping has complex behavior in comparison to the selected subdomain approximation or where the training data points are distant from one another. Furthermore, when the
amount of data is large or the actual mapping is not complex, using boundary fuzziness does not improve
the performance and it is better to use nonfuzzy partitions to reduce the number of parameters needed to define
an MC-HARP approximation. A good index for measuring the effect of fuzziness is the difference between
the values of deviation measures $\sigma_F$ for fuzzy partitions, $\gamma=1$, and nonfuzzy partitions, $\gamma=0$. If $\sigma_F$ significantly decreases by setting $\gamma$ to be one instead of zero, one should use fuzzy partitions. One should keep in
mind that the performance measures like $\sigma_F$ and $RMS_F$ are average values for the entire input domain and
do not represent the local performance of the constructed mappings. Hence, in the case of nonhomogeneous,
actual mappings, one can improve performance by using fuzziness for regions of the input domain where the
actual mapping is complex or data points are sparse. By partitioning the input domain into a few subregions
and calculating the change in the value of the deviation measure $\sigma_F$ for these subregions of the input domain
using fuzzy subdomains instead of nonfuzzy subdomains, one can apply fuzziness more efficiently for the
nonhomogeneous mappings.

Figure 3.6 shows the variation of performance and complexity indices with respect to the amount of training
data for fuzzy $\mathcal{P}_1^H$ and $\mathcal{P}_3^H$ schemes for selecting splitting thresholds. It is evident that for a given number
of data, the MC-HARP approximations built using the $\mathcal{P}_1^H$ scheme have slightly lower approximation error,$RMS_F$ and $e_{\text{max}}$ and approximation deviation, $\sigma_F$, than the approximations built using the $\mathcal{P}_3^H$ scheme. By
increasing the amount of training data, this difference between performances decreases and for large data sets,

Fig. 3.6 Variation of the performance and complexity indices with respect to the amount of
data for different schemes for selecting splitting thresholds
both $\mathcal{P}^1_H$ and $\mathcal{P}^3_H$ schemes develop approximations with the same performance. On the other hand, as shown in Fig. 3.6, the MC-HARP approximations built using $\mathcal{P}^1_H$ scheme require smaller numbers of subdomains and parameters than the MC-HARP approximations built using $\mathcal{P}^3_H$ scheme. This observation indicates that the $\mathcal{P}^1_H$ scheme develops approximations that are significantly simpler and exhibit slightly better performance than approximations using the $\mathcal{P}^3_H$ scheme. The main reason for this characteristic is that, the $\mathcal{P}^1_H$ scheme selects splitting thresholds such that to balance the numbers of training data in the offspring subdomains of a typical intermediate subdomain of a HARP partition. Hence, the $\mathcal{P}^1_H$ scheme increases the probability of developing large subdomains in the final partition of a HARP training process in comparison to $\mathcal{P}^3_H$. The $\mathcal{P}^1_H$ scheme randomly selects splitting thresholds and therefore increases the probability of developing small subdomains in the intermediate partitions and consequently the final partition of a HARP partitioning tree. In other words, the deterministic $\mathcal{P}^1_H$ scheme develops partitioning trees that are more balanced and have a smaller number of final subdomains than the random $\mathcal{P}^3_H$ scheme. For the sake of approximation simplicity, the $\mathcal{P}^1_H$ scheme is recommended for selecting splitting thresholds.

To study the combined effects of the MC-HARP fuzziness and schemes for selecting splitting thresholds on the performance of MC-HARP, we use the two-dimensional XOR function as the actual mapping. The XOR function is simple, it is nonlinear, and it can be considered as the magnified behavior of any two-dimensional mapping for a subset of four neighboring data points. Hence, in this numerical example, the results represent the behavior of an MC-HARP approximation for four neighbor data points when the number of superimposed HARP approximations is large; i.e., $p=2000$. Figure 3.7 shows the variation of $\sigma_F$ with respect to the sample size $p$ for four MC-HARP solutions built for the XOR problem using fuzzy $\mathcal{P}^1_H$, nonfuzzy $\mathcal{P}^2_H$, nonfuzzy $\mathcal{P}^2_H$, and nonfuzzy $\mathcal{P}^3_H$ schemes for selecting splitting thresholds and defining subdomain boundaries. It is evident that the fuzzy $\mathcal{P}^1_H$ has the lowest deviation measure and the nonfuzzy $\mathcal{P}^2_H$ has the highest deviation measure. We expect this behavior because the $\mathcal{P}^1_H$ collection of partitions is the smallest and the $\mathcal{P}^2_H$ collection of partitions is the largest collection of the HARP collections $\mathcal{P}^1_H$, $\mathcal{P}^2_H$, and $\mathcal{P}^3_H$. The smaller the collection is, the smaller the deviation measure is. On the other hand, the MC-HARP approximation using the $\mathcal{P}^1_H$
scheme reaches its steady deviation measure for larger sample size than the MC-HARP approximation using \( \mathcal{P}_H^2 \). Because the \( \mathcal{P}_H^1 \) collection is smaller than the \( \mathcal{P}_H^3 \) collection, its partitions are more distinct, and consequently, the probability that the newly added HARP approximation to the sample of partitions is different from the previously added HARP approximations to the sample is much higher for the \( \mathcal{P}_H^1 \) collection than the \( \mathcal{P}_H^2 \) collection. The MC-HARP fuzziness decreases the approximation deviation. The fuzzy \( \mathcal{P}_H^1 \) has a \( \sigma_F \) measure smaller than the nonfuzzy \( \mathcal{P}_H^1 \), as shown in Fig. 3.7.

Figure 3.8 presents the behavior of the MC-HARP approximations \( \bar{F}(x) \) and their corresponding deviation functions \( \sigma(x) \) constructed using different partitioning schemes. The solutions corresponding to the fuzzy \( \mathcal{P}_H^1 \) and nonfuzzy \( \mathcal{P}_H^3 \) are smoother than the solutions corresponding to the nonfuzzy \( \mathcal{P}_H^1 \) and \( \mathcal{P}_H^3 \). As we previously mentioned, the behavior of an MC-HARP solution to the XOR problem is a good representation of the behavior of an MC-HARP solution to any mapping problem for a small subset of its corresponding data set. In other words, the MC-HARP solution to the XOR problem represents the magnified behavior of the MC-HARP solution over a small region of the input domain. One would expect the deviation function \( \sigma(x) \) to behave like a tent-shaped function between training data points with stakes located at training data points; i.e., \( \sigma \) is small for training data points. Furthermore, the expected microscopic smoothness developed by the MC-HARP approximation for regions between data points is typically shown in Fig. 3.8.

Table 3.1 shows the maximum deviation measure \( \sigma_{\text{max}} \) and the average deviation measure \( \sigma_F \) for the test data points and also the number of parameters for the MC-HARP approximations using different partitioning schemes. The MC-HARP approximations using the deterministic \( \mathcal{P}_H^1 \) scheme require more parameters and are more complex than the MC-HARP approximations using random \( \mathcal{P}_H^2 \) and \( \mathcal{P}_H^3 \) schemes, as shown in Table 3.1. The reason for this observation is that with the \( \mathcal{P}_H^1 \) scheme and the constant subdomain approximation \( \theta \), a final partition with subdomains containing an equal number of data points should have four subdomain. The partitioning tree corresponding to the \( \mathcal{P}_H^1 \) scheme always has one intermediate partition with two subdomains each containing two data points. However, the partitioning tree corresponding to the random schemes \( \mathcal{P}_H^2 \) and \( \mathcal{P}_H^3 \) can have an intermediate partition with two subdomains where one of them contains one data point and the other one contains three data points. Further partitioning of the subdomain with three data points may lead to the development of two subdomains where one of them contains one data point and the other one contains two diagonal data points, for example \((0,0,0)\) and \((1,1,0)\), that can be fit by a constant \( \theta \). Thus, with the random schemes \( \mathcal{P}_H^2 \) and \( \mathcal{P}_H^3 \), final partitions with three subdomains may exist in the MC-HARP sample of HARP partitions. Consequently the number of parameters in the corresponding MC-HARP approxi-

<table>
<thead>
<tr>
<th>partitioning scheme</th>
<th>( \sigma_{\text{max}} )</th>
<th>( \sigma_F )</th>
<th>( K )</th>
</tr>
</thead>
<tbody>
<tr>
<td>fuzzy ( \mathcal{P}_H^1 )</td>
<td>0.368</td>
<td>0.318</td>
<td>4.0</td>
</tr>
<tr>
<td>nonfuzzy ( \mathcal{P}_H^1 )</td>
<td>0.500</td>
<td>0.423</td>
<td>4.0</td>
</tr>
<tr>
<td>nonfuzzy ( \mathcal{P}_H^2 )</td>
<td>0.500</td>
<td>0.440</td>
<td>3.8</td>
</tr>
<tr>
<td>nonfuzzy ( \mathcal{P}_H^3 )</td>
<td>0.500</td>
<td>0.473</td>
<td>3.8</td>
</tr>
</tbody>
</table>

Table 3.1 Deviation and complexity measures for MC-HARP solutions for the XOR problem
Fig. 3.8 MC-HARP solutions for the XOR problem
mations should be smaller than four because some of the HARP approximations require only three constant functions to fit the XOR data.

3.2.5 Effect of the Dimensionality of the Input Domain

Data points in higher dimensional spaces are very sparse. The higher the dimensionality of the input domain is, the sparser and more spread apart are the data points. This phenomenon is known as the curse of dimensionality, a phrase due to Bellman (1961). To illustrate this sparseness, consider a uniform distribution of data points in a ten-dimensional unit hypercube. A subcube covering 10 percent of the range of each coordinate contains \((0.1)^{10}\) of the data and thus is practically empty. On the other hand, to contain 10 percent of the data, a subcube should cover almost 80 percent of the range of each coordinate!

Besides the sparseness of data in higher dimensional spaces, there are several other curses of dimensionality. Increasing the dimension of the input domain can cause adverse effects such as greater computational costs, slower convergence to the actual mapping, less robustness, and worse generalization. Furthermore, according to the theory of Kolmogorov n-widths, there is no linear function space with dimension of smaller order than \(n^d\) that achieves the rate of approximation \(1/n\), where \(d\) is the dimension of the input space, \(n^d\) is the number of parameters in the constructed mapping, and the approximation error is the integral of the squared error between the actual mapping and the constructed mapping over the input domain. Traditional polynomials, splines, and trigonometric functions are examples for a linear space of functions. Thus approximations that are linear with respect to their parameters use exponentially many parameters \(o(n^d)\) to achieve approximation rates of order \(o(1/n)\). This exponential explosion of the number of required parameters is also a curse of dimensionality.

In this numerical simulation, we want to study the performance of the MC-HARP method with respect to the dimension of the input domain and the amount of data. We use the multi-dimensional function \(G_t\) to generate training and test data. The \(G_t\) function can be described by

\[
G_t(x; d) = \prod_{j=1}^{d} Tri(x_j)
\]

where \(x_j\) is the \(j\)th coordinate of the vector \(x\), \(d\) is the dimension of the input domain, and the univariate, triangle-shaped function \(Tri(z)\) is defined as follows

\[
Tri(z) = \begin{cases} 
0 & \text{if } z < 0 \text{ or } 1 < z \\
2z & \text{if } 0 \leq z \leq 0.5 \\
-2z + 2 & \text{if } 0.5 \leq z \leq 1
\end{cases}
\]

and is shown in Fig. 3.9.

The \(Tri(z)\) function does not have continuous higher derivatives. Hence, the function \(G_t(x; d)\) does not have continuous higher derivatives and has Lipschitz order of one. Approximating nonsmooth mappings requires more data and is more vulnerable to the curse of dimensionality than approximating smooth mappings.
Since the support of the function $\text{Tri}(z)$ is the unit interval $[0,1]$, the support of the $G_t(x;d)$ function is a $d$-dimensional unit hypercube. We use a uniform random number generator to select training and test data points in this unit hypercube. The test set contains 4000 data points uniformly distributed. The number of uniformly distributed training data points varies between 100 to 100000. We choose the subdomain approximation $\theta$ to be a linear function. The tolerance value $\varepsilon$ for the termination criterion is set to be 0.001. The sample size $p$ is 20. The fuzzy $\mathcal{P}^1_H$ scheme is used to select splitting thresholds for the one-dimensional $G_t(x;1)$ and the fuzzy $\mathcal{P}^1_H$ scheme is used for $G_t(x;d)$ where $d$ is greater than one.

Figure 3.10 shows the variation of performance and complexity indices for MC-HARP approximations with respect to the amount of training data $N$ for different dimensions $d$ of the input domain. The approximation error, $\text{RMS}_F$ and $e_{\text{max}}$, and the approximation deviation, $\sigma_F$, decreases as the number of data increases. The rate of convergence depends on the dimension of the input domain and the size of the data set. For small data sets, the rate of convergence is slow and is almost zero for higher dimensional data. When the amount of data is larger than a critical value, the performance indices start to significantly decay and their rates of convergence increase. The higher the dimension of the input domain is, the larger this critical amount of data is. By presenting more data, the rate of convergence slows down and the performance indices become steady or oscillate, as shown in Fig. 3.10 for $d$ equals to one or two. The lower bound of the performance indices for large data sets depends on the termination tolerance $\varepsilon$. The smaller is the tolerance value $\varepsilon$, the smaller are the lower bounds of the performance indices.

To compare rates of convergence quantitatively, we plot in Figs. 3.10(a) through 3.10(c) a dotted line representing a specific rate of convergence. The dotted line in Figs. 3.10(a) and 3.10(b) represent the ideal rate of order $o(1/\sqrt{N})$ for the $\text{RMS}_F$ measure and in Fig. 3.10(c), it represents the ideal rate of order $o(1/\sqrt{N/\ln N})$ for the maximum error. (The decaying exponent for the ideal rate of convergence for both $\text{RMS}_F$ and $e_{\text{max}}$ is $-1/2$, but for $e_{\text{max}}$ we compute the rate using $N/\ln N$ instead of $N$ due to the suggestion by Stone (1982)). For the deviation measure $\sigma_F$, the dotted line is similar to the $\text{RMS}_F$ measure.

It is evident that for $d=1$ and $d=2$, the rate of convergence is even greater than the ideal rate. For higher dimensional data, the rate of convergence becomes equal to the ideal rate for large data sets. It can be concluded that the ultimate rate of convergence for an MC-HARP approximation is independent of the dimension of the input domain. However the amount of data required to reach the ultimate rate of convergence directly depends on the dimensionality of the data set. The reason for this characteristic is that, in order to converge
to the actual mapping, a data set should have adequate data points to represent the main features of the actual mapping at different regions of its input domain. Therefore, a minimum number of data points should exist at different regions of the input domain to represent the main features of the actual mapping in those regions. The convergence to the actual mapping starts as we supply the minimum required data. The sparseness due to the curse of dimensionality increases the amount of data needed to supply the required minimum number of data at different regions of the input domain.

The Me-HARP method does not have any control over the sparseness of data caused by the curse of dimensionality. However, in so far as we supply adequate data, the MC-HARP approximation converges to the actual mapping with a rate that is independent from the dimensionality of data. Our numerical simulation shows that the rate of convergence for the MC-HARP approximation is $o(1/\sqrt{N})$ for the $RMS_F$ performance measure and $o(1/\sqrt{N/\ln N})$ for the $\epsilon_{max}$ performance measure. The approximation deviation $\sigma_F$ of an MC-HARP approximation also have the same ultimate rate of convergence as the $RMS_F$ measure, as shown in Fig. 3.10(b).

Figure 3.10(d) shows the variation of the number of parameters, $\overline{K}$, in the MC-HARP approximations with respect to the amount of training data for different dimensions of the input domain. The dotted line in Fig. 3.10(d) represents the line $\overline{K}=N$. It is evident that the number of parameters $\overline{K}$ is smaller than the number of data points for any size of data set and dimension of the input domain. As we also showed in Fig. 3.2, by increasing the amount of data, the increasing rate of $\overline{K}$ decreases and curves $\overline{K}(N)$ saturate for large
amounts of data. The upper bound of $K(N)$ directly depends on the termination tolerance $\varepsilon$ indicating the complexity of an MC-HARP approximation is a function of $\varepsilon$. The smaller is the tolerance value $\varepsilon$, the larger is the upper bound of $K(N)$. For the lower dimensional data sets, the saturation of $K(N)$ happens for a smaller amount of data than it does for the higher dimensional data sets. Since the number of parameters in an MC-HARP approximation is smaller than the number of data points for different dimensions of the input domain and the ultimate rate of convergence for an MC-HARP approximation does not depend on the dimensionality of the data set, an MC-HARP approximation does not suffer of the exponential explosion of parameters caused by the curse of dimensionality for linear spaces of approximations.

As can be observed in Fig. 3.10, the approximation error $RMS_{F}$ starts to significantly decrease when the amount of data is large enough. For any dimension of the input domain, there is a critical amount of data that for data sets containing more data than this critical amount, the approximation error significantly decays. This critical amount of data can be used as a quantitative measure for the adequacy of data. Data sets with more data than this critical amount contain enough data to adequately represent the main features of the actual mapping. The problem with using the $RMS_{F}(N)$ curve for measuring the adequacy of data is that often we do not have enough data to hold out for a test set. Fortunately, based on our numerical simulations, the deviation curve $\sigma_{F}(N)$ has the same behavior as $RMS_{F}(N)$, as shown in Fig. 3.10(b). The measure $\sigma_{F}$ can always be computed for a set of test data points without knowing their corresponding target outputs. A data set contains an adequate amount of data if its corresponding rate of decrease for $\sigma_{F}(N)$ is significant. The adequacy of the amount of data depends on the dimensionality of the data set. The minimum amount of data needed to represent the main features of the actual mapping is controlled by the sparseness caused by the curse of dimensionality. The adequate amount of data which is defined by the decreasing rate of the measure $\sigma_{F}(N)$, is definitely greater than the required minimum amount of data and depends on the dimensionality of data. In next chapters, we will establish a framework for measuring the minimal amount of data and the adequacy of data.

A reasonable way of comparing the efficiency of different consistent approximations, besides their rates of convergence and the complexity of their constructed approximations, is the amount of data required by these approximations to have the same accuracy. Here we want to compare the HARP and MC-HARP approxi-
mations. As we mentioned, these two approximations are consistent. Figure 3.11 shows the variation of performance indices for a typical HARP approximation for the $G_t$ function with respect to the amount of training data for different dimensions of the input domain. It is evident that, for a given amount of data, an MC-HARP approximation has a better rate of convergence and higher accuracy than the HARP approximation for different dimensions of the input domain, indicating that the superposition of a sample of HARP approximations has better efficiency and accuracy than each individual HARP approximation on average. This characteristic can be mathematically shown as follows: By defining the measure of accuracy to be the $L_q$ norm of the error between the actual mapping $G$ and the constructed mapping, the bias between a typical HARP approximation $F_{c,i}$ and the actual mapping $G$ takes the form

$$e_{F_{c,i}} = \int_{\mathbb{D}} |F_{c,i}(x) - G(x)|^q \, dx \quad (3.13)$$

Similarly for the MC-HARP approximation $\bar{F}$, the approximation bias is defined as

$$e_{\bar{F}} = \int_{\mathbb{D}} |\bar{F}(x) - G(x)|^q \, dx \quad (3.14)$$

**Proposition:** The average of approximation errors for a sample of HARP approximations is greater than the approximation error of their sample mean average. In other words,

$$\frac{1}{P} \sum_{i=1}^{P} e_{F_{c,i}} \geq e_{\bar{F}} \quad (3.15)$$

where $\bar{F}(x) = \frac{1}{P} \sum_{i=1}^{P} F_{c,i}(x)$.

**Proof:** Using Eqns. (3.13) and (3.14), the following equation can be written

$$\frac{1}{P} \sum_{i=1}^{P} e_{F_{c,i}} - e_{\bar{F}} = \frac{1}{P} \sum_{i=1}^{P} \int_{\mathbb{D}} |F_{c,i}(x) - G(x)|^q \, dx - \int_{\mathbb{D}} \left| \frac{1}{P} \sum_{i=1}^{P} F_{c,i}(x) - G(x) \right|^q \, dx \quad (3.16)$$

and can be simplified as follows

$$\frac{1}{P} \sum_{i=1}^{P} e_{F_{c,i}} - e_{\bar{F}} = \int_{\mathbb{D}} \left[ \frac{1}{P} \sum_{i=1}^{P} e_i(x)^q - \frac{1}{P^q} \sum_{i=1}^{P} e_i(x) \right]^q \, dx \quad (3.17)$$

where $e_i(x) = F_{c,i}(x) - G(x)$. Now, by Holder's inequality, the integrand of the the integral in Eqn. (3.17) is always positive. Therefore, the integral is positive. Consequently the difference in the left hand side of Eqn. (3.17) should be positive. Hence, the inequality (3.15) holds for $1 \leq q < \infty$. ■
The above proposition holds for any approximation, including, a HARP approximation. The proposition indicates that the mean average approximation has smaller bias than the mean average of biases for the sample of approximation $s$. It can be concluded that, for a selected point in the input domain, there may exist an approximation in the constructed sample of approximations that has smaller bias than the sample mean average approximation. However, for the HARP approximator, the probability of there existing a HARP approximation $F_C$ in the sample of HARP approximations with a smaller bias than the MC-HARP approximation $F$ for all points in the input domain is small because of the randomness in the HARP partitioning scheme; i.e., $\Pr(|F_C(x) - G(x)| < |F(x) - G(x)| \forall x \in \mathcal{D})$ is small. Actually, based on our numerical simulations, the probability of there existing a HARP approximation $F_C$, in the sample of approximations built by MC-HARP, with smaller bias $e_F$ than the bias of the MC-HARP approximation $e_F$ is also small; i.e., $\Pr(e_{F_C} < e_F)$ is small. This observation indicates that averaging of HARP approximations through MC-HARP improves the approximation accuracy. The MC-HARP approximation converges to the actual mapping more uniformly than do HARP approximations. The probability of there existing a HARP approximation with better pointwise or overall performance than the MC-HARP approximation may increase as the amount of data increases. For large data sets, the sample size of MC-HARP is set to be small due to computational cost and similar performance of HARP approximations and their mean average approximation.

3.3 MC-HARP and Neural Networks

We have developed the MC-HARP method to construct approximations that belong to the collection $\mathcal{L}_\mathcal{G}$. The multivariate functions in $\mathcal{L}_\mathcal{G}$ can be represented using scalar summation, scalar multiplication, and composition of a finite collection of univariate functions. These three fundamental operations with the space of univariate functions develop the collection $\mathcal{L}_\mathcal{G}$ containing the MC-HARP and HARP approximations. Furthermore, as we discuss in Appendix A, these fundamental operations are also done by a standard sigma-pi processing unit in a neural network to compute its output. The operational similarities between a processing neuron and an $\mathcal{L}_\mathcal{G}$ function raise the idea of representing $\mathcal{L}_\mathcal{G}$ functions by neural networks.

In this section, we will show how HARP and MC-HARP approximations can be represented by neural networks and consequently can be implemented on neural hardwares and computers. One can generalize from the neural representation of a HARP approximation to represent any $\mathcal{L}_\mathcal{G}$ function by a neural network. Furthermore, since the final product of MC-HARP can be represented as a neural network, the MC-HARP method develops an environment for simultaneously building and training mapping neural networks. Hence, MC-HARP can be used for all applications of mapping neural networks. The MC-HARP environment builds parsimonious network architectures with good generalization. The independence of subdomain approximations and of HARP partitions make the MC-HARP training process highly parallelizable. Therefore, the MC-HARP environment not only builds data-based mappings that take advantage of the massively parallel architecture of neural networks but it also has a highly parallel training strategy enabling it to exhibit good speed up and scalability on multiprocessor computers.

In this section, we show how an $\mathcal{L}_\mathcal{G}$ function can be represented by a neural network. Next we present the neural network representation of a HARP approximation and the growth of a HARP neural network during
the HARP training process. Finally, we show that an MC-HARP approximation can be modeled as a modular neural network.

3.3.1 Neural Network Representation of an $\mathcal{L}_J$ Function

A standard sigma-pi unit computes its net input by a scalar summation of a finite collection of products of its inputs. The unit imposes a nonlinear, univariate output function on its net input to compute its output. Hence, based on the definition of an $\mathcal{L}_J$ function, if the inputs of a sigma-pi unit are computed by $\mathcal{L}_J$ functions, its output is an $\mathcal{L}_J$ function. Consequently, a neural network of sigma-pi units with a single output unit represent an $\mathcal{L}_J$ function because the inputs of the network are the independent variables that are univariate and definitely belong to $\mathcal{L}_J$. Furthermore, a sigma-pi unit represents the main computational block of an $\mathcal{L}_J$ function since it is capable of performing the fundamental operations of addition, multiplication, and composition of univariate functions. Therefore an $\mathcal{L}_J$ function can be represented by a neural network of sigma-pi units with a single output unit. A neural network of sigma-pi units represents a unique $\mathcal{L}_J$ function but the neural network representation of an $\mathcal{L}_J$ function may not be unique. In other words, an $\mathcal{L}_J$ function may be represented by different network architectures.

Figure 3.12 shows four multivariate functions and their corresponding neural representations. In this figure, the value shown inside each unit is its bias. A bias is the constant part of the net input. We do not assign any bias value to input units to indicate that they do not do any computation and just transfer the external signal to other units. The univariate function whose variable is the net input $\Omega$, shown nearby a processing unit, is the output function of that unit. Functions on Fig. 3.12 can be represented by network architectures different from those shown in this figure. For example, the function $H_4$ can be represented by a network with

$$H_1(x_1, x_2) = w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_5x_2^2$$

$$H_2(x_1, x_2) = \frac{w_0 + w_1x_1 + w_2x_2}{w_3 + w_4x_1 + w_5x_2}$$

$$H_3(x_1, x_2, x_3) = x_1^3 - x_2^3 + x_3^3$$

$$H_4(x_1, x_2) = x_1x_2$$

Fig. 3.12 Neural network representations of $\mathcal{L}_J$ functions
a single sigma-pi output unit whose output function is the identity function. However, the network configurations for $H_3$ and $H_4$ just use standard sigma units (i.e., units whose net inputs are weighted summations of their inputs) and do not multiply outputs of their processing units. One can generalize from these examples that an $L_9$ function can be represented by a neural network of sigma units and the multiplication can be replaced by summation and composition. Here we illustrate two approaches for this computational replacement.

The multiplication of a finite number of values can be represented as follows

$$\prod_{j=1}^{n} x_j^{w_j} = e^{\sum_{j=1}^{n} w_j \ln x_j}$$  \hspace{1cm} (3.18)

when $x_j$ is negative, $\ln x_j = \ln |x_j| + i\pi$ which is a complex value and the network handles its computation in the complex domain. Durbin and Rumelhart (1989) used Eqn. (3.18) to develop product units for backpropagation networks. We use Eqn. (3.18) to build the neural representation of the $H_3$ function in Fig. 3.12.

Another alternative for replacing multiplication with summation and composition of univariate functions is to use the following equality

$$x_1x_2 = 0.5 \left[ \cosh(\sinh^{-1}x_1 + \sinh^{-1}x_2) - \cosh(\sinh^{-1}x_1 - \sinh^{-1}x_2) \right]$$  \hspace{1cm} (3.19)

The neural network representation of the right hand side of Eqn. (3.19) is shown in Fig. 3.12 for the function $H_4$. The multiplication of a finite number of values can be done by grouping them into pairs, using Eqn. (3.19) to compute the product of pairs, and repeating the whole process for the products of pairs.

Hence a neural network with only sigma units can handle the multiplication of a finite number of values using Eqn. (3.18) or (3.19). The technique shown in Eqn. (3.18) for replacing multiplication with summation and composition requires a smaller number of processing units than the technique shown in Eqn. (3.19). However, the approach in Eqn. (3.19) can handle negative numbers while the approach in Eqn. (3.18) requires complex computations.

The above techniques show that $L_9$ functions can be represented by scalar summation and composition of a finite collection of univariate functions and the scalar multiplication is a redundant operation for defining $L_9$. Furthermore, $L_9$ functions can be represented by a neural network of sigma units.

### 3.3.2 Neural Network Representation of A HARP Approximation

The functional representation of a HARP approximation $F_C$ is shown in Eqn. (2.42). When the partitioning function $\psi$ and the subdomain approximation $\theta$ belong to the collection $L_9$, the HARP approximation $F_C$ is an $L_9$ function. Hence, $F_C$, as an $L_9$ function, can be represented by a neural network. The HARP approximation is a summation of a finite collection of local subdomain approximations. The characteristic function $\chi$ for each subdomain defines the support of its corresponding subdomain approximation $\theta$. As
shown in Eqn. (2.42), the HARP approximation $F_C$ is a summation whose summand has two main parts: an approximation part and a partitioning part. The approximation part is represented by the squashed subdomain approximation $\Phi(\theta)$. The partitioning part, representing the characteristic function, is computed by a product of a finite number of gated partitioning functions $\widetilde{R}(\psi)$. Therefore, the global configuration of a HARP approximation is a summation of approximations localized by their corresponding characteristic functions. This global configuration can be considered as a neural network with one hidden layer and one sigma-pi output unit, as shown in Fig. 3.13. The weights of the connections between the hidden layer and the output unit are all unity. In a HARP neural network the hidden layer consists of two block of units. One block of hidden units, called the partitioning block, computes characteristic functions and the other block, called the approximation block, computes subdomain approximations. The computational process of these two blocks are executed in parallel. The propagation of the outputs of these blocks is channeled by the pattern of connectivity between the hidden layer and the output unit in a way that each characteristic function gates its corresponding subdomain approximation, as shown in Fig. 3.13.

The approximation block of a HARP neural network contains groups of hidden units whose outputs represent the squashed subdomain approximation $\Phi(\theta)$. The subdomain approximation $\theta$ can be a neural network or can be represented by a neural network. For example, $\theta$ belongs to the collection $\mathcal{L}_G$ and, as we showed, there exists a neural network representation for every $\mathcal{L}_G$ function. The squashing function $\Phi$ is the output function for the output unit of the neural network representing $\theta$. The squashing function shown in Fig. 2.2 is one of the common forms of output functions shown in Fig. A.4 in Appendix A. The approximation units

![Fig. 3.13 Global configuration of a HARP neural network](image-url)
shown in Fig. 3.13 in the approximation block of a HARP neural network are actually neural networks representing subdomain approximations $\theta$ and their outputs are squashed by the squashing function $\Phi$. The parameters $w$ of a subdomain approximation $\theta(x; w)$ represents the weights associated with the connections of the neural network for $\theta$.

The partitioning block of a HARP neural network contains hidden units whose outputs represent the gated partitioning functions $\tilde{\Gamma}(\psi)$. The multiplication of the outputs of hidden units in the partitioning block that is based on the hierarchical relationships of intermediate subdomains in a HARP partitioning tree computes the characteristic functions $\tilde{\chi}$. Equation (2.41) shows the hierarchical pattern of the product of gated functions $\tilde{\Gamma}(\psi)$ for computing the characteristic function $\tilde{\chi}$. When $\psi$ belongs to the collection $L_0$, it can be represented by a neural network. The output of the neural network computing $\psi$ is gated by the gate function $\tilde{\Gamma}$ as the output function for its output unit. For example, the simple linear partitioning function shown in Eqn. (2.34) can be represented by a single sigma unit. Consequently, the gated, linear partitioning function $\tilde{\Gamma}(\psi(x; c))$ is computed by a hidden sigma unit whose output function is $\tilde{\Gamma}$ shown in Eqn. (2.36) and Fig. 2.9 and its coefficient $c$ are the weights associated with the connections between the input layer and the corresponding hidden unit for $\tilde{\Gamma}(\psi(x; c))$. The splitting thresholds $d$ are parameters for defining the gate function $\tilde{\Gamma}$ as shown in Eqn. (2.36).

The architecture of a HARP neural network is shown in Fig. 3.14 with more detail than Fig. 3.13. The hidden layer contains two blocks of units. One block contains subdomain approximation units whose outputs are $\Phi(\theta)$ and the other block contains partitioning units whose outputs are $\tilde{\Gamma}(\psi)$. The partitioning and approximation units in a HARP neural network might contain groups of neural processing units in order to represent

Fig. 3.14 A HARP neural network
ψ and θ functions. The output unit is a sigma-pi unit. The pattern of connectivity between the hidden layer and the output unit is dictated by the hierarchical relationships among subdomains in a HARP partitioning tree. The weights for connections between the hidden layer and the output unit are all unity. The weights associated with the connections between the input layer and the hidden layer represent the parameters c and w in the partitioning function ψ and the subdomain approximation θ, respectively.

The functional representation of a HARP solution for the XOR problem defined in Section 2.5 takes the form

\[ F_C(x) = \Gamma(\psi(x; c); -\infty, \frac{3}{16})x_2 + \Gamma(\psi(x; c); \frac{3}{16}, +\infty)(-x_2 + 1) \]  

(3.20)

where the gate function \( \Gamma \) is defined in Eqn. (2.31) and ψ is the linear partitioning function whose parameters c are \((0.5, -0.125)\). The HARP neural network for the HARP approximation \( F_C(x) \) in Eqn. (3.20) is shown in Fig. 3.15 with its corresponding partitioning tree and partition of the input domain. The boundary between subdomains \( \mathcal{D}_1 \) and \( \mathcal{D}_2 \) is the line \( \psi(x; c) = 0.5x_1 - 0.125x_2 = 3/16 \). The units in the hidden layer of the HARP neural network are standard sigma processing units. The subdomain approximations \( \theta_1(x) = x_2 \) and \( \theta_2(x) = -x_2 + 1 \) are not squashed by \( \Phi \).

The membership characteristic function \( \check{X} \) developed by HARP is a positive-valued function and is computed by the product of a finite number of gated partitioning functions \( \tilde{\Gamma}(\psi) \) that are positive, as shown in Eqn. (2.41). Hence using the relation (3.18), this product can be computed by summation and composition of \( \tilde{\Gamma}(\psi) \) functions as follows

\[ \check{X}_{\mathcal{D}_1}(x; \gamma) = e^{\sum_{\psi} \ln \tilde{\Gamma}(\psi(x; c_j); d_j; \gamma)} \]  

(3.21)

![Fig. 3.15 A HARP solution for the XOR problem](image-url)
where all the parameters are defined in Eqn. (2.41). Furthermore, since the value of the squashed subdomain approximation $\Phi(\theta)$ might be negative, the multiplication of a squashed approximation $\Phi(\theta)$ and its corresponding characteristic function $X_{g1}$ can be computed using the relation (3.19). Therefore, the HARP approximation $F_C$ shown in Eqn. (2.42) can be computed using summation and composition of a finite collection of univariate functions, if the partitioning function $\psi$ and the subdomain approximation $\theta$ can be computed the same way. For example, the large collection of $\mathcal{L}_{g2}$ functions can be represented using only scalar summation and composition of univariate functions. It can be concluded that the HARP neural network shown in Fig. 3.14 can be reconfigured using only sigma units. Assuming $\psi$ and $\theta$ can be represented by the sigma units, the configuration of a HARP neural network with only sigma units is shown in Fig. 3.16. The blocks of hidden units in this configuration of the HARP neural network computes the characteristic function $X$ using Eqn. (3.21) and multiply $X$ by the squashed subdomain approximation $\Phi(\theta)$ through Eqn. (3.19) using the neural network configuration shown in Fig. 3.12 for the $H_4$ function.

3.3.3 HARP training and Neural Networks

The HARP training process starts with the input domain as the parent subdomain and fits the selected subdomain approximation $\theta$ to the data in the parent subdomain. If the fit is not acceptable then partition the parent subdomain and fit $\theta$ to the offspring subdomains. The training process continues this scheme until the termination criterion is satisfied for all subdomains. Hence, the HARP algorithm increases the complexity of a HARP approximation during the training process. Consequently, the HARP training process allocates new computational units for the HARP neural network as the training approximation error decreases. In other words, a HARP neural network grows during the training process.

To illustrate the growing process for a HARP neural network, Fig. 3.17 and 3.18 show two consecutive configurations, $F_1$ and $F_2$, of a HARP neural network during the training process. Their corresponding partitioning trees and splitting trees are also shown. For the sake of clarity, we choose the configuration of a HARP neural network with sigma-pi units shown in Fig. 3.14 instead of the configuration with sigma units shown.

![Fig. 3.16 A HARP neural network with only sigma units](image-url)
in Fig. 3.16. The generated subdomains are sequentially numbered from one where the subdomain number
one is the input domain; i.e., \( D_1 = \mathbb{D} \). The \( F_1 \) partition has four subdomains called \( D_4 \) through \( D_7 \) and their corresponding subdomain approximations are \( \theta_4 \) through \( \theta_7 \), respectively. Furthermore, all units are number-
ered such that common units between \( F_1 \) and \( F_2 \) neural networks have the same tag number. The tag number
of each unit is written beside it in Figs. 3.17 and 3.18.

A partitioning unit computes the gated partitioning function \( \tilde{T}(\psi) \) and its output function is the gate func-
tion \( \tilde{T} \). We use the convention that for the \( i \)th split of the splitting tree, the value \( \tilde{T}(\psi_i; -\infty, d_i) \) corre-
sponds to the left offspring subdomain in the partitioning tree. For example, the subdomain \( D_5 \) is generated by the
splits 1 and 2 as shown in Fig. 3.17(c). Since \( D_5 \) is the right offspring of \( D_2 \) which is the left offspring of the
input domain \( D_1 \), the characteristic function of \( D_5 \), \( \tilde{X}_5 \), is the product of \( \tilde{T}(\psi_2; d_2, +\infty) \) and
\( \tilde{T}(\psi_1; -\infty, d_1) \), as shown in Fig. 3.17(a). Hence, the pattern of connectivity among outputs of partitioning
units is dictated by the hierarchical ordering of the partitioning tree. The weights associated with the connec-

![Diagram](image-url)

Fig. 3.17 Intermediate HARP neural network \( F_1 \)
(a) Network configuration, (b) Partitioning tree, (c) Splitting tree

73
tions between a partitioning unit corresponding to the $i$th split and the input layer represent the parameters of the partitioning function $\psi_i(x; c)$. For example, the weight vectors $w_{o2}$ and $w_{o3}$ are equal and represent the parameters of $\psi_2$ that are randomly determined.

An approximation unit computes the squashed subdomain approximation $\Phi(\theta)$. The weights associated with the connections between an approximation unit and the input layer represents the parameters of $\theta(x; w)$. The weights $w$ are computed through a parameter estimation algorithm as discussed in Section 2.2. The selected parameters estimation algorithm can be a neural network learning algorithm, if $\theta$ can be represented

---

**Fig. 3.18 Intermediate HARP neural network $F_2$**

(a) Network configuration, (b) Partitioning tree, (c) Splitting tree

74
by a neural network or can be an optimization-based learning algorithm. Hence, one can use learning algorithms for neural networks to adjust weights associated with the connections between an approximation unit and the input layer, for example $w_{07}$ for the $F_1$ neural network.

The hierarchical ordering of an intermediate HARP partitioning tree dictates the pattern of connectivity of an intermediate HARP neural network during the HARP training process. Furthermore, the HARP partitioning randomly selects the parameters of partitioning functions associated with the HARP splitting tree. Assuming that HARP does subdomain approximation subdomain by subdomain, during a subdomain parameter estimation stage of the HARP training process, for example parameter estimation for $\theta_5$ of $F_1$, the configuration of the intermediate HARP neural network, for example $F_1$, is fixed and all connections have assigned weights except for connections corresponding to the subdomain approximation whose parameters are currently estimated, for example the weight vector $w_{08}$ for $\theta_5$ as shown in Fig. 3.17(a). Hence, the weight updating during the training process in a HARP neural network is localized. Unlike traditional neural networks that simultaneously updates all their weights during a training iteration, a HARP network only adjusts the weights corresponding to those subdomain approximations whose subdomains contain the presented training patterns.

During the training process HARP first computes predicted outputs for the presented training patterns. Then, HARP computes the error between the predicted outputs and expected outputs. If the training error is not acceptable, HARP subdomain training process adjusts the weights of those subdomain approximations whose subdomains contain the presented training patterns. The partitioning block of a HARP neural network determines which subdomains contains some training data points by computing the values of subdomain characteristic functions for all presented data points. If the value of a subdomain characteristic function $\chi_{\delta_5}(x)$ for a data point $x$ is zero, it means that the subdomain $\delta_5$ does not contain $x$ and $\chi_{\delta_5}(x)=1$ indicates that $x$ belongs to $\delta_5$.

When HARP cannot reduce the training error by adjusting the parameters of subdomain approximations, HARP partitions poorly approximated subdomains. At the stage of subdomain partitioning, the HARP neural network grows and allocates new computational units. Figure 3.18 shows the configuration of the intermediate HARP neural network $F_2$ as a consecutive configuration for the intermediate HARP neural network $F_1$ shown in Fig. 3.17. In Fig. 3.18(a), the dashed connections and units represent newly created elements that are added to the $F_1$ network to build the $F_2$ network. The HARP training process generates the $F_2$ configuration from the $F_1$ configuration assuming that subdomain approximation $\theta_5$ of $F_1$ poorly fits data in the subdomain $\delta_5$. Consequently, the HARP partitioning randomly splits $\delta_5$ to two offspring subdomains $\delta_8$ and $\delta_9$, as shown in Fig. 3.18(b). The partitioning function $\psi_4$ corresponding to the split number 4, as shown in Fig. 3.18(c), is developed with random parameters to split $\delta_5$ around the splitting threshold $d_4$. To compute the characteristic functions for $\delta_8$ and $\delta_9$, HARP allocates two new partitioning units, 12 and 13, and connects them to the pattern of connectivity of the partitioning block of $F_1$ using the hierarchical ordering of the $F_2$ partitioning tree. Furthermore, HARP duplicates the approximation unit 8 into the approximation unit 14. The approximation units 8 and 14 in the $F_2$ network computes squashed subdomain approximation $\Phi(\theta_8)$ and $\Phi(\theta_9)$, respectively and may have initial weight values equal to the adjusted weights for $\theta_5$; i.e., the initial
values for $w_{08}$ and $w_{014}$ of the $F_2$ network are equal to $w_{08}$ of the $F_1$ network. During the partitioning stage of the HARP training process, the $F_1$ network grows to become the $F_2$ network.

The HARP training process follows a sequence of subdomain partitioning and subdomain training stages to simultaneously build and train a HARP neural network. The subdomain partitioning stage grows the constructed network and allocates new partitioning and approximation units. The subdomain training stage adjusts the connection weights of approximation units. The partitioning stage is computationally fast because the parameters of the new partitioning functions are randomly determined and splitting thresholds are determined with simple computations. The pattern of connectivity of a HARP neural network is dictated by its partitioning tree. A HARP neural network, besides having the fundamental components of a neural network (i.e., the pattern of connectivity, propagation rule, and learning rule), also has a growing process.

3.3.4 Neural Network Representation of an MC-HARP Approximation

The functional representation of an MC-HARP approximation $\bar{F}$ and its deviation measure $\sigma$ are shown in Eqn. (2.45). If the HARP approximations $F_C$ can be represented by neural networks, $\bar{F}$ and $\sigma$ have neural network representations because $\bar{F}$ and $\sigma$ can be represented as the scalar summation of the HARP approximations and their squared distances from their mean. The MC-HARP approximation $\bar{F}$ and its deviation measure $\sigma$ can be modeled by a modular neural network, as shown in Fig. 3.19. The hidden network of an MC-HARP neural network consists of $p$ HARP neural networks. The basic module of a modular MC-HARP neural network is a HARP neural network. The output units compute statistical indices including sample mean average and standard deviation for the sample of $p$ outputs predicted by the HARP neural networks. The HARP modules are independent from one another, so they can be built and run in parallel.

During the MC-HARP training process, the weights associated with the connections between output units of the HARP modules and statistical output units of the MC-HARP neural network do not change. Only the weights associated with the HARP networks are adjusted using the HARP training process. The constant and equal weights assigned to the outputs of the HARP modules indicates that the modular MC-HARP neural net-

![Fig. 3.19 MC-HARP neural network](image)
work is similar to a committee of trained experts; HARP modules whose responses for each input pattern are equally plausible because they have been trained on the same data set and have the same structure. The independence of subdomain approximations and of HARP modules makes the MC-HARP training process highly parallelizable.

3.4 Conclusions

In this chapter, we have studied the performance of an MC-HARP approximation through numerical simulations. By increasing the sample size for the MC-HARP method, the performance and complexity measures of an MC-HARP approximation becomes steady like a typical Monte Carlo method. The approximation error decays as the sample size increases and converges to a bias value between the expected approximation of the random HARP approximations and the actual mapping. Furthermore, the deviation measure of an MC-HARP approximation increases as the sample size increases and converges to the standard deviation of the HARP approximations. Based on our numerical simulations, we have recommended use of the sample size which, for the MC-HARP approximations with samples larger than this size, the deviation measure does not change significantly. This size can be taken as the minimum sample size for MC-HARP.

The approximation error and the deviation measure of an MC-HARP approximation decreases as the amount of data increases, indicating that the MC-HARP approximation is consistent. Furthermore, the complexity of an MC-HARP approximation increases as the amount of data increases and saturates for large data sets. The performance of the MC-HARP approximation with different subdomain approximations leads us to the recommendation that one should always use simple subdomain approximations unless using more complex subdomain approximations improves the performance, reduces the complexity of the constructed MC-HARP approximation, or a priori knowledge about the mapping approximation problem dictates the complexity of the subdomain approximation.

For a given data set, the HARP algorithm controls the size of subdomains in any region of the input domain based on the relative complexity of the actual mapping and the selected subdomain approximation in that region in comparison with the selected tolerance value for the subdomain training process. HARP partitioning is approximation-error-driven. Hence, for a selected tolerance value, HARP attempts to effectively distribute subdomains in the input domain to fit the data within the selected tolerance. Consequently, HARP partitions, HARP approximations, and MC-HARP approximations are functions of the selected tolerance value. By increasing the tolerance value, the complexity of an MC-HARP approximation decreases and its approximation error increases. Furthermore, for tolerance values close to zero, the deviation measure increases as the tolerance value increases until it reaches its maximum value and then decreases to zero for large tolerance values.

The boundary fuzziness of HARP approximations improves the performance of an MC-HARP approximation for regions of the input domain where the actual mapping has complex behavior in comparison to the selected subdomain approximation or the training data points are sparse. Furthermore, when the amount of data is large or the actual mapping is not complex, it is better to use nonfuzzy partitions to reduce the number of parameters needed to define an MC-HARP approximation. The $\mathcal{P}_n^H$ scheme for selecting splitting thresh-
olds develops approximations that are significantly simpler and exhibit slightly better performance than the approximations using $\mathcal{P}_2^H$ and $\mathcal{P}_3^H$ schemes.

Data points in higher-dimensional spaces are very sparse and MC-HARP does not have any control over the sparseness of data caused by the curse of dimensionality. However, for adequate, noise-free data sets, the MC-HARP approximation converges to the actual mapping with a rate that is independent of the dimensionality of data. The ultimate rate of convergence for the approximation error of an MC-HARP approximation is $o(1/\sqrt{N})$, where $N$ is the number of training data points. The amount of data required to reach the ultimate rate of convergence directly depends on the dimensionality of the data set. Furthermore, for a given amount of data, an MC-HARP approximation has a better rate of convergence and higher accuracy than a HARP approximations, indicating that the superposition of HARP approximations through MC-HARP improves the approximation accuracy and the MC-HARP approximation converges to the actual mapping more uniformly than HARP approximations.

In this chapter, we have also shown that any $\mathcal{L}_d$ function can be represented by a neural network. Consequently, HARP and MC-HARP approximations that are $\mathcal{L}_d$ functions have neural network representations and can be implemented on neural hardwares. An MC-HARP approximation can be modeled by a modular neural network whose basic module is a HARP neural network. Furthermore we have shown that $\mathcal{L}_d$ functions can be represented only by scalar summation and composition of a finite collection of univariate functions. Hence $\mathcal{L}_d$ functions including HARP and MC-HARP approximations can be modeled by neural networks with only sigma units.

Since the final product of MC-HARP can be represented as a neural network, the MC-HARP method develops an environment for simultaneously building and training mapping neural networks. The HARP and MC-HARP training processes are highly parallelizable and can exhibit good speed up on multiprocessor computers. The pattern of connectivity of a HARP neural network is dictated by its partitioning tree. A HARP neural network, besides the fundamental components of connectivity pattern, propagation rule, and learning rule, also has a growing process. The subdomain partitioning stage of a HARP training process grows the constructed network and allocates new partitioning and approximation units. The weight updating of a HARP training process is localized. Unlike the traditional neural networks that simultaneously updates all their weights during a training iteration, a HARP neural network only adjusts the weights corresponding to those subdomain approximations whose subdomains contain the presented training patterns.
CHAPTER FOUR
Performance of MC-HARP for Noisy Data

"The great tragedy of science: the slaying of a beautiful hypothesis by an ugly fact."

Thomas Huxley

“What can we know? or what can we discern, when error chokes the windows of the mind?”

Sir John Davies

We have established procedures and rules for selecting the main components of the MC-HARP method namely, the subdomain approximation $\theta$, the number of partitions $p$, the subdomain partitioning schemes, the continuity and boundedness modifications, and the neural network representation. In this chapter, we provide a concept for selecting the tolerance value $c$ for the termination criterion for subdomain training process of MC-HARP. The proposed concept defines a criterion for selecting the optimal size of partitioning trees of the approximations built by the HARP algorithm in order to construct a parsimonious approximation by the MC-HARP strategy that has the best generalization for a given set of noisy data.

In the HARP algorithm, the distribution of subdomains is related to the distribution of data points and the complexity of the actual mapping. For a given data set, the HARP algorithm controls the size of subdomains in any region of the input domain based on the relative complexity of the data behavior with respect to the selected subdomain approximation $\theta$ in that region in comparison with the selected tolerance. Because the HARP partitioning is approximation-error-driven, for a selected tolerance value, HARP attempts to effectively distribute subdomains in the input domain based on the regional complexity of the data behavior. Consequently, we can use the tolerance value $c$ to control the distribution, size, and number of subdomains in a HARP partition. In other words, the tolerance for the approximation error can control an approximation-error-driven partitioning scheme and its corresponding local approximation. Therefore, for a given data set, we can control the complexity of an MC-HARP approximation $\hat{F}$ by the tolerance value $c$ and the structural details of $\hat{F}$, including the distribution, size, and number of subdomains, are automatically determined by the HARP algorithm.

For noisy data, the complexity of a data-based approximate mapping is a critical factor influencing approximation accuracy. If the constructed approximation is too simple, then it does not have enough adaptivity to capture main features of the actual mapping and consequently is biased. On the other hand a too complex approximation interpolates noisy data and is dominated by noise, therefore it misses main features of data destroyed by noise. Hence both simple and complex approximations have low approximation accuracy and poor generalization. The dependence of approximation accuracy on the approximation complexity and quali-
ty of data demonstrates the importance of selecting the tolerance value in order to build an MC-HARP approximation with good generalization using a noisy data set.

In this chapter we will use numerical simulations to study performance of the MC-HARP method for noisy data. We will define performance indices to investigate the complexity-dependent accuracy of an MC-HARP approximation for different noise amplitudes and amounts of data. General trends in performance of constructed MC-HARP approximations are extracted to establish a framework for performance analysis of MC-HARP and also a model selection criterion. Furthermore we will show that there exists an optimal tolerance value corresponding to an approximation with the optimal complexity and lowest approximation error.

4.1 What is a Good Approximation?

To provide a general development, let us assume that a vector of noisy target values (observations) \( y(\eta) = [y_1(\eta) \ y_2(\eta) \ ... \ y_N(\eta)]^T \) has been obtained from the following model

\[
y_k(\eta) = G(x_k) + \lambda \eta_k \quad k = 1, ..., N
\]

where \( G \) is an unknown, multivariate mapping, \( \mathcal{G} = \{x_k\}_{k=1}^N \) is the set of (training) data points in the input domain \( \mathcal{D} \), \( G(x_k) \) is the unknown target value for the \( k \)th data point, \( N \) is the number of (training) data points, \( \eta = [\eta_1 \ \eta_2 \ ... \ \eta_N]^T \) is a vector of zero mean, independent, identically distributed random noise with variance equals to one, and \( \lambda \) is the amplitude of the noise, and by construction, also the standard deviation of the noise. We will refer to \( \mathcal{D}(\eta) = \{(x_i, y_i(\eta))\}_{i=1}^N \) as the (noisy, training) data set.

For a given data set \( \mathcal{D} \) and a fixed subdomain approximation \( \theta \), changing the tolerance value \( \varepsilon \) for the training (data fitting) process at each subdomain generates a family of random partitions \( \mathcal{C}(\varepsilon) \). Corresponding to each partition, there is an approximation \( F_{\mathcal{C}(\varepsilon)} \). We refer to the collection \( \{F_{\mathcal{C}(\varepsilon)} \forall \varepsilon \geq 0 \} \) as the family of approximations \( \mathcal{G} \). Assuming a fixed number of partitions \( p \), the MC-HARP algorithm generates a family of approximations \( \mathcal{G} \) defined as follows

\[
\mathcal{G} = \{F(\varepsilon) : F(\varepsilon) = \frac{1}{p} \sum_{i=1}^{p} F_{\mathcal{C}(\varepsilon)} \forall \varepsilon \geq 0 \}
\]

(From now on \((-)^{-}\) is used to indicate the expected value of a variable with respect to \( \mathcal{C} \) and can be approximated by the sample mean average of \( p \) values of the variable.)

The number of subdomains \( s_{\mathcal{C}}(\varepsilon) \) in each partition \( \mathcal{C} \) is a function of \( \varepsilon \). When \( \varepsilon \) is large enough \( s_{\mathcal{C}} \) is equal to one, meaning that \( \theta \) can fit the data with accuracy \( \varepsilon \) and thus there is no need to partition the input domain. By setting \( \varepsilon \) equal to zero, \( s_{\mathcal{C}} \) reaches its maximum value because we want the approximations \( F_{\mathcal{C}} \) to fit the data perfectly (interpolation). A simple subdomain approximation function \( \theta \) can fit data perfectly when the subdomains are small enough. The size of the smallest subdomain in \( \mathcal{C} \), \( \delta_{\mathcal{C}} \) is directly related to the number of subdomains in \( \mathcal{C} \). The \( E_{\mathcal{C}}[\delta_{\mathcal{C}}] \) value decreases as \( E_{\mathcal{C}}[s_{\mathcal{C}}] \) increases. For a perfect fit (\( \varepsilon = 0 \)), \( \delta_{\mathcal{C}} \) reaches its minimum value and consequently the number of subdomains \( s_{\mathcal{C}} \) reaches its maximum value. Therefore \( s_{\mathcal{C}}(\varepsilon) \) is a nonincreasing function of \( \varepsilon \) whose maximum and minimum values are \( s_{\mathcal{C}}(0) \) and one, respectively.
The approximation family $\mathcal{F}$ is a collection of functions $F_{C(\varepsilon)}$ that are different from one another in the number of subdomains $s_{C}(\varepsilon)$. At $\varepsilon=0$, $F_{C}$ perfectly fits the data with the maximum number of subdomains in $C$ and when $\varepsilon$ is equal to a large value, there is only one subdomain in $C$, the input domain, and $F_{C}$ is represented by the subdomain approximation $\theta$. Therefore by changing the value of $\varepsilon$ from zero to a large value, the approximation $F_{C(\varepsilon)}$ moves from a local interpolation $F_{C(0)}$ to a global parametric approximation $\theta$. The same characteristics are true for $\bar{F}(\varepsilon)$ as the average of a sample of local approximations $\{F_{C(\varepsilon)}\}^{p}_{i=1}$. Thus, $\varepsilon$ is an indicator of the complexity of the approximations $F_{C}$ and $\bar{F}$. We want to select the optimal (best) approximation from the family $\mathcal{F}$ built by the MC-HARP algorithm (model selection problem). In other words, we want to establish a mathematical basis for selecting the optimal (best) tolerance value $\varepsilon^{*}$ that gives a model with optimal (best) complexity. (From now on $^{*}$ indicates that a variable is optimal in some sense).

In our problem statement about model selection in $\mathcal{F}$, the word “best” or “optimal” has been left without a precise definition. There is obviously no way to formulate a solution without being more specific. We define the risk in approximating $G$ by $\bar{F}$ as follows

$$RISK_{\bar{F}} = \left[ \int_{\mathcal{D}} [\bar{F}(x; \mathcal{I}(\eta)) - G(x)]^{2} dx \right]^{1/2} \quad (4.3)$$

where $\mathcal{D}$ is the input domain and $\mathcal{I}(\eta)$ is the set of noisy training data. When $x$ does not have a uniform distribution over $\mathcal{D}$, the integral in Eqn. (4.3) should be computed based on the distribution of $x$. Since $\bar{F}$ is built using the noisy data set $\mathcal{I}(\eta)$, therefore $\bar{F}$ is a function of noise and consequently $RISK_{\bar{F}}$ is a random variable. To make the risk quantity more mathematically tractable, we compute its expected value as follows

$$ERISK_{\bar{F}} = \int_{N} \left[ \int_{\mathcal{D}} [\bar{F}(x; \mathcal{I}(\eta)) - G(x)]^{2} dx \right]^{1/2} d\mathcal{Q}(\eta) \quad (4.4)$$

where $\mathcal{Q}(\eta)$ is the probability density function for the random noise vector $\eta$ with support $\mathcal{N}$. For a normally distributed noise vector, $\mathcal{N}$ is equal to $\Re^{N}$. The discrete form of $ERISK_{\bar{F}}$ is defined as

$$ERISK_{\bar{F}} = E_{\eta} \left[ \left[ \frac{1}{N_{t}} \sum_{x \in \mathcal{S}_{t}} [\bar{F}(x; \mathcal{I}(\eta)) - G(x)]^{2} \right]^{1/2} \right]^{2} \quad (4.5)$$

where $\mathcal{S}_{t} = \{x_{k}\}^{N_{t}}_{k=1}$ is a (test) set of $N_{t}$ data points in $\mathcal{D}$ and $E_{\eta}$ is an operator representing the expected value with respect to the noise vector $\eta$. The set $\mathcal{S}_{t}$ is generally disjoint from the set $\mathcal{S}$ of training data points.

Another performance measure that is closely related to $ERISK$ is the root mean squared error of prediction $ERMS_{F}^{Test}$ defined as

$$ERMS_{F}^{Test} = E_{\eta}[RMS_{F}^{Test}] \quad (4.6)$$

where
where \( N_t \) is the number of data in the test set \( \mathcal{T}_t \), \( x \) belongs to a set of testing data points \( \mathcal{G} \) that is disjoint from the set of training data points \( \mathcal{G} \) and the random error \( \eta' \) added to the noisy testing target values \( y(\eta') \) in Eqn. (4.7) are uncorrelated with the random errors \( \eta \) added to the target values of the noisy training data set \( \mathcal{T}_t \). \( \text{ERMS}_{\text{est}} \) is a combination of two measures, the bias between the constructed approximation \( \mathcal{F} \) and the actual mapping \( \mathcal{G} \) represented by \( \text{ERISK}_{\mathcal{F}} \) and the standard deviation of noise \( \lambda \). Thus, an approximation \( \mathcal{F} \) that minimizes the \( \text{ERISK}_{\mathcal{F}} \) measure will also minimize \( \text{ERMS}_{\text{est}} \) and vice versa. To compute \( \text{ERMS}_{\text{est}} \), we need to know the noisy target values at testing data points. The common practice is to divide the data set into independent training and test sets, use the training set to build an approximation, and study its performance over the test set. A measure \( \text{ERMS}_{\text{train}} \) corresponding to the training set can also be defined using Eqns. (4.6) and (4.7) and replacing the test set \( \mathcal{T}_t \) with the training set \( \mathcal{T} \).

### 4.2 How to Select the Optimal Approximation

For a given data set, an approximation is optimal if it has the lowest \( \text{ERISK} \) value. Here we define optimality in the family of approximations \( \mathcal{F} \). An approximation \( \mathcal{F}(\varepsilon^*) \) is an optimal approximation of \( \mathcal{G} \) if it has the lowest \( \text{ERISK} \) value in \( \mathcal{F} \). We refer to each partition corresponding to the optimal approximation as an optimal partition and to the tolerance value \( \varepsilon^* \) corresponding to the optimal approximation as the optimal tolerance value. Assuming \( \mathcal{F}(\varepsilon^*) \) is an optimal approximation we can write

\[
\text{ERISK}_{\mathcal{F}(\varepsilon^*)} \leq \text{ERISK}_{\mathcal{F}(\varepsilon)} \quad \forall \varepsilon \geq 0 \tag{4.8}
\]

and the set of optimal partitions is \( \{ \mathcal{C}_j(\varepsilon^*) \}_{j=1}^p \). The proposed model selection criterion in \( \mathcal{F} \) is based on the risk in approximation. Unfortunately, \( \text{ERISK} \) cannot actually be computed without knowing \( \mathcal{G} \). Thus, in practice, we need an estimate of \( \text{ERISK} \) computed from the data and then minimize this estimate with respect to \( \varepsilon \) to obtain an estimate of the optimal approximation for \( \mathcal{G} \). There are a variety of criteria which have appeared in the literature for data driven model selection. Here we discuss some famous criteria and then explain our proposed approach.

A popular technique for model selection is to minimize \( \text{RMS}_{\text{est}} \mathcal{F}(\varepsilon) \) for selecting the best model. Here, we use \( \text{RMS}_{\text{est}} \mathcal{F}(\varepsilon) \) as an index replacing \( \text{ERISK}_{\mathcal{F}} \) because for large test sets, minimizing \( \text{RMS}_{\text{est}} \mathcal{F}(\varepsilon) \) leads to the same optimal model as minimizing \( \text{ERISK}_{\mathcal{F}} \). The problem with this technique is that often there are enough data to build an adequate approximation, but not enough to hold out for a test set. The situation becomes worse in higher dimensional mapping problems. This problem motivates statisticians to search for techniques that reuse data to compute the performance measure.

Perhaps the most commonly used performance estimating technique for model selection is cross-validation (CV) \( (\text{CV}) \) \( \text{ (Stone 1974, Geisser 1975, Efron 1979)} \). In the cross-validation approach, one partitions the data set \( \mathcal{T} \) into \( J \) disjoint subsets \( \{ \mathcal{T}_v \}_{v=1}^J \) usually with an equal amount of data in each. One then uses the data set
set $\mathcal{T} - \mathcal{T}_v$ to build an approximation $\hat{F}_v$ and computes the mean squared error of $\hat{F}_v$ for the set $\mathcal{T}_v$. This process is repeated for all subsets $\{ \mathcal{T}_v \}_{v=1}^J$. Then, an estimate for the prediction error $ERMS_{\hat{F}(\epsilon)}^{Test}$ is defined as follows

$$CV_J(\epsilon) = \left[ \frac{1}{J} \sum_{v=1}^J \frac{1}{N_v} \sum_{(x,y) \in \mathcal{T}_v} [\hat{F}_v(x; \epsilon) - y]^2 \right]^{\frac{1}{2}}$$

(4.9)

where $N_v$ is the number of data points in $\mathcal{T}_v$. $CV_J(\epsilon)$ is called the $J$-fold cross-validation root mean squared error and selection of $\epsilon$ through minimizing $CV_J(\epsilon)$ is called the $J$-fold cross-validation approach. When in each trial of the cross-validation approach, only one data point is hold out of the training process, the approach is called leave-one-out cross-validation.

Another model selection technique reusing data to compute an estimate for the prediction error is the bootstrap method (Efron 1982, Efron and Tibshirani 1986). In bootstrapping, one assumes equal probabilities for all data points in the data set $\mathcal{T}$ of size $N$ and draws $B$ samples of size $N$, $\{ \mathcal{T}_v \}_{v=1}^B$, from $\mathcal{T}$ through a Monte Carlo algorithm. The drawing of each data point is independent from other drawings and is done with replacement meaning that a data point may be selected several times. The number of unduplicated data in a selected subset $\mathcal{T}_v$ probably is smaller than $N$. One then builds an approximation $\hat{F}_v$ using the data set $\mathcal{T}_v$ and computes the mean squared error of $\hat{F}_v$ for the data set $\mathcal{T}$. This process is repeated for all subsets $\{ \mathcal{T}_v \}_{v=1}^B$. Then, an estimate for the prediction error is calculated as follows

$$BS_B(\epsilon) = \left[ \frac{1}{B} \sum_{v=1}^B \frac{1}{N} \sum_{(x,y) \in \mathcal{T}} [\hat{F}_v(x; \epsilon) - y]^2 \right]^{\frac{1}{2}}$$

(4.10)

The bootstrap sample size $B$ should be large enough ($B \rightarrow \infty$) to compute a useful estimate of the prediction error. Minimization of $BS_B(\epsilon)$ with respect to $\epsilon$ is the bootstrap technique for model selection. Combining different techniques in the sampling theory leads to a number of variations of these model selection criteria resampling data, namely the generalized cross-validation, stratification, and jackknife methods (Weiss and Kulikowski 1991).

The model selection criteria based on resampling techniques are computationally expensive. For example, one needs to construct $J$ approximations in the $J$-fold cross-validation method for each value of tolerance $\epsilon$ or build $B$ approximations in the bootstrap method. Also, in these methods, an approximation $\hat{F}$ is built using a subset of data set. Holding out some data from the training process reduces the performance of the constructed approximations and therefore adds bias to the estimate of the prediction error. These shortcomings have motived researchers to develop model selection techniques using all of the given data to build a model and to compute an estimate for the prediction error. These techniques are generally developed in the information theory and widely used for multivariate function approximation problems. A popular criterion in this category is the root predicted squared error (Barron 1984), defined as
\[ RPS(\varepsilon) = \left[ \frac{1}{N} \sum_{(x,y) \in \mathcal{F}} [\overline{F}(x, \varepsilon) - y]^2 + \frac{2\mathcal{K}(\varepsilon)}{N}\lambda^2 \right]^{1/2} \]  

(4.11)

where \( \mathcal{F} \) is the given data set, \( N \) is the number of data points, \( \lambda^2 \) is the variance of noise added to the target values, and \( \mathcal{K}(\varepsilon) \) is the number of adjustable parameters in the approximation \( \overline{F}(\varepsilon) \). \( RPS(\varepsilon) \) is an estimate for \( RMS_{\overline{F}(\varepsilon)}^{\text{Test}} \) and can be minimized to select the optimal tolerance \( \varepsilon^* \). \( RPS \) is the square root of two terms as shown in Eqn. (4.11): the mean squared error of the training set that is minimized during the training process and a term that penalizes the constructed approximation based on the number of free parameters (as a measure of complexity). The \( RPS \) performance criterion prevents building an overly complex approximation that accurately fits the training data but has poor generalization. The main shortcoming of the \( RPS \) criterion is that it needs a good estimate of the noise variance which is usually unknown. If the estimate of the noise variance is too much less than its actual value \( \lambda^2 \), the approximation will tend to be too complex. When \( \lambda^2 \) is unknown, one can compute a rough estimate for it using the variance of target values. A better estimate for \( \lambda^2 \) is \( (N/(N-\mathcal{K})) RMS_{\overline{F}}^{\text{Train}})^2 \) where \( \overline{F} \) is a simple approximation with a small number of parameters \( \mathcal{K} \). 

In the literature, one can find other criteria in the class of the predicted squared error namely, the \( c_p \) statistic (Mallows 1973), the minimum description length (Rissanen 1983), the conditional prediction error (Thompson 1978), and the Akaike information criterion (Akaike 1973).

In the following sections we will use numerical simulations to study the characteristics of the MC-HARP method for noisy data. We will define performance indices to study how the performance of an MC-HARP approximation depends on its complexity for different noise amplitudes and numbers of data. We will extract the general trends in the performance of the MC-HARP approximations to establish a new criterion for model selection in the approximation family \( \mathcal{F} \).

### 4.3 Performance Analysis through Numerical Simulations

We use a one dimensional function \( G_{r1} \) and a two dimensional function \( G_{r2} \) to generate training and test data. The function \( G_{r2} \) has been defined in Eqn. (2.43) and the function \( G_{r1} \), with domain \([0, 1]\), can be described by

\[
G_{r1} = \begin{cases} 
1. & 0.0 \leq x \leq 0.2 \\
-5x + 2 & 0.2 \leq x \leq 0.4 \\
0. & 0.4 \leq x \leq 0.6 \text{ and } 0.8 \leq x \leq 1.0 \\
400(x - 0.6)^2 & 0.6 \leq x \leq 0.7 \\
400(x - 0.8)^2 & 0.7 \leq x \leq 0.8 
\end{cases} 
\]  

(4.12)

and is shown in Fig. 4.1. The \( G_{r1} \) function is a continuous function with the same characteristics as the \( G_{r2} \) function, i.e. in a picturesque way, both functions are a combination of a ramp and a peak. Both functions are nonhomogeneous in the sense that they represent different mathematical relationships between their in-
puts and outputs in different parts of their input domains. The maximum value of $G_{r1}$ is 4.0 and the maximum value of $G_{r2}$ is one. We select the training and test data points on regular grids, i.e. data points are spread uniformly and with equal distance from one another over the input domain. The number of test data $N_t$ for the $G_{r1}$ function is 1000 and for $G_{r2}$ is 1936 (44 x 44). The set of training data points $\mathcal{S}$ is chosen such that the number of common points in $\mathcal{S}$ and the set of test points $\mathcal{S}_t$ is zero or is small in comparison with $N_t$. In our numerical simulations the noisy training target values $\mathcal{Y}$ are generated by adding random noise to the actual values of functions as shown in Eqn. (4.1). The random noise has a standard normal distribution and three different amplitudes of noise 0.05, 0.3, and 1.0 are used to generate noisy training data sets.

The subdomain approximation $\theta$ is chosen to be a linear function that has the flexibility to reduce its number of parameters whenever the number of data in a subdomain is smaller than its number of parameters. A term truncation scheme, as described in Section 2.2.4, is used to downsize $\theta$ for small subdomains. For the data fitting (training) process at each subdomain, we choose the sum of the squared residuals as the fitness index and since the subdomain approximation is linear in its parameters, we compute its parameters explicitly by least squares. A ramp squashing function with $\mu=0$, as described in Section 2.2.5, is composed on the subdomain approximation $\theta$. We accept a subdomain approximation when its maximum training residual is less than a tolerance value $\varepsilon$. For the subdomain partitioning process, we use the linear partitioning function, defined in Eqn. (2.34), with $r=2$. For selecting the splitting thresholds, we choose the fuzzy $\mathcal{P}_3^\varepsilon$ scheme for the $G_{r1}$ function and the fuzzy $\mathcal{P}_1^\varepsilon$ scheme for the $G_{r2}$ function.

4.4 Performance Indices

To study the MC-HARP method and to find trends in its performance, we use five different performance indices, $ERISK_{\mathcal{F}}, ERMS_{\mathcal{F}}^{Train}, E\sigma_{\mathcal{F}}^{Train}, E\sigma_{\mathcal{F}}^{Test}$, and $ERPS_{\mathcal{F}}$. The risk of approximation $ERISK_{\mathcal{F}}$ is defined in Eqn. (4.5) and is computed for the test set. The root mean squared error of the training set $ERMS_{\mathcal{F}}^{Train}$ is computed using Eqns. (4.6) and (4.7) for the training data. The standard deviation of approximations $F_{\mathcal{C}}$ about their mean $\bar{F}$ computed for the training set, $E\sigma_{\mathcal{F}}^{Train}$, is defined as

$$E\sigma_{\mathcal{F}}^{Train} = E_{\eta} \left[ \frac{1}{N} \sum_{x \in \mathcal{S}} E_{\mathcal{C}} \left[ F_{\mathcal{C}}(x; \mathcal{F}(\eta)) - \bar{F}(x; \mathcal{F}(\eta)) \right]^2 \right]^{1/2} \right]$$

(4.13)
where $N$ is the number of training data, $\mathcal{T}(\eta)$ is the set of noisy training data, and $\mathcal{E}$ is the set of training data points. A similar index $E\sigma^2_{\text{est}}$ can be defined for the test set by replacing $\mathcal{E}$ and $N$ in Eqn. (4.13) with $\mathcal{E}_t$ and $N_t$. In Eqn. (4.13) the standard deviation of the random values $F_c(x)$ is approximated by the standard deviation of a sample of size $p$ of them. $\bar{F}$ is the approximation constructed by the MC-HARP method that is the sample mean average of $p$ random approximations $F_c$ built by the HARP method. In our numerical simulations, the sample size $p$ is set to be 30. The MC-HARP method approaches the actual mapping by building a sample of approximations constructed by the HARP method, the index $ERISK_F$ represents the bias between the mean average of this sample $\bar{F}$ and the actual mapping $G$ and $E\sigma_F^2$ represents the variation of these solutions around their mean. The root predicted squared error $ERPS_F$ is computed for the training set and is defined using Eqn. (4.11) as follows

$$ERPS_F = E_\eta \left[ (RMS_{F_{\text{train}}}^2 + \frac{2K}{N} \lambda^2) \right]^{\frac{1}{2}} \quad (4.14)$$

where $RMS_{F_{\text{train}}}^2$ is computed using Eqn. (4.7) with $\mathcal{T}$ instead of $\mathcal{E}$, $\lambda$ is the amplitude of noise and $K$ is the sample mean average of the numbers of parameters $K_c$ in the $p$ approximations $F_c$. The number of parameters $K_c$ in an approximation $F_c$ is equal to the sum of numbers of parameters in all subdomain approximations $\theta$ used to build $F_c$.

The motivation for using definition (4.14) for $ERPS_F$ is as follows: $ERPS_F$ represents the expected value of the root predicted square error of $\bar{F}$, $RPS_F$, with respect to the noise vector. $\bar{F}$ is the mean average of a sample of approximations $F_c$. Using Eqn. (4.11), on can compute $RPS$ for each $F_c$ and then calculate the expected value of $RPS_{F_c}$ with respect to $C$ as follows

$$E_C[RPS_{F_c}^2] = E_C \left[ (RMS_{F_c}^2 + \frac{2K_c}{N} \lambda^2) \right] \quad (4.15)$$

which can be written as

$$E_C[RPS_{F_c}^2] = E_C \left[ (RMS_{F_c}^2) + \frac{2\lambda^2}{N} E_C[K_c] \right] \quad (4.16)$$

By replacing the first term in the right hand side of Eqn. (4.16) with $(RMS_{F_{\text{train}}}^2)^2$ which is smaller and using $K$ for $E_C[K_c]$, Eqn. (4.16) defines $RPS_F$. From now on we refer to $\bar{K}$ as the number of parameters in the approximation $\bar{F}$ built by the MC-HARP method.

We calculate the expected value with respect to the noise vector $E_\eta$ used in the definitions of the performance indices by averaging over a sample of nine sets of noisy training target values. At each trial of our simulation, we add a noise vector $\eta$ to the actual function values computed at a given training data points to generate a set of noisy training target values $y(\eta)$. We then use the generated data to build an approximation $\bar{F}$ and calculate the defined performance indices for it. We repeat this process nine times and compute the
mean average of the performance indices calculated at these nine trials to estimate their expected values with respect to the noise vector.

4.5 Performance Curves for MC-HARP Approximations

For a fixed subdomain approximation \( \theta \) and a fixed number of partitions \( p \), the approximation \( \bar{F} \) constructed by the MC-HARP method is a function of the quantity of the given data, its own complexity, and quality of the given data. The indicator for the amount of data is the number of training data \( N \), for the complexity of \( \bar{F} \) is the tolerance value \( \varepsilon \), and for the quality of data is the noise amplitude \( \lambda \). Consequently, the performance indices \( E_{\text{RISK}}(\bar{F}), E_{\text{RMSE}}(\bar{F}), E_{\text{RMS}}(\bar{F}), E_{\text{RPS}}(\bar{F}) \) are all functions of \( N, \varepsilon, \) and \( \lambda \) and we show them as functions of these three variables, for example \( E_{\text{RISK}}(\bar{F})(N, \varepsilon, \lambda) \). In our numerical simulations we want to study the behavior of these indices with respect to the amount of data, the complexity of the constructed approximations, and the amplitude of noise. For a fixed value of \( N \) and \( \lambda \), these performance indices are only function of \( \varepsilon \). One should notice that all the performance indices are always positive.

Figures 4.2 through 4.5 show performance curves for the actual functions \( G_{r_1} \) and \( G_{r_2} \) regarding different numbers of data and noise amplitudes. We shall study the behavior of these curves. The most important performance curve is the approximation risk \( E_{\text{RISK}}(\bar{F}) \) that cannot be produced for the actual data. By increasing the value of \( \varepsilon \) from zero to a large number, \( E_{\text{RISK}}(\bar{F}) \) generally starts to decrease, reaches its minimum at \( \varepsilon = \varepsilon^* \) and then increases. The optimal model built by the MC-HARP method for a given data set is \( \bar{F}(\varepsilon^*) \) whose approximation risk is \( E_{\text{RISK}}(\bar{F})(N, \varepsilon^*, \lambda) \).

The \( E_{\text{RMSE}}(\bar{F}) \) index is equal to zero when \( \varepsilon \) is zero because we accurately fit the training data (local interpolation). The performance index \( E_{\text{RMSE}}(\bar{F}) \) is a nondecreasing function with respect to \( \varepsilon \) and reaches its maximum value when \( \varepsilon \) is a large number (\( \varepsilon_{\text{max}} \)). At \( \varepsilon = \varepsilon_{\text{max}} \) the number of subdomains \( s \) in the partitions \( C \) is equal to one and we build a parametric approximation equals to \( \bar{F} \). For the tolerance values greater than or equal to \( \varepsilon_{\text{max}} \), the number of subdomains in the random partitions \( C \) are all equal to one and all the approximations \( F \) and their mean \( \bar{F} \) are equal to \( \bar{F} \). Therefore, none of the performance indices changes for \( \varepsilon \geq \varepsilon_{\text{max}} \). We refer to the interval \([0, \varepsilon_{\text{max}}]\) as the domain of the performance curves.

The standard deviation index \( E_{\text{RMS}}(\bar{F}) \) is equal to zero at \( \varepsilon = \varepsilon_{\text{max}} \) because there is no deviation among the approximations \( F \); i.e., all of them are equal to \( \theta \). Also, at \( \varepsilon = 0 \), the performance index \( E_{\text{RMS}}(\bar{F}) \) is equal to zero because at each training data point \( \mathbf{x} \in \mathcal{X} \), we force the approximations \( F \) to have the same value as the noisy target value \( y \) corresponding to \( x \). Therefore, for each \( \mathbf{x} \in \mathcal{X} \), there is no deviation in the values of \( F(x) \) computed for a sample of random partitions \( C \); i.e., \( \forall (\mathbf{x}, y) \in \mathcal{X} \ F_c(x) = y \ \forall C \). Consequently, using Eqn. (4.13) leads to \( E_{\text{RMS}}(\bar{F}) \) equal to zero when \( \varepsilon = 0 \). Since \( E_{\text{RMS}}(\bar{F}) \) is a nonconstant, positive function with respect to \( \varepsilon \), and is equal to zero at the boundaries of its domain, it should have at least one maximum point at some tolerance value between zero and \( \varepsilon_{\text{max}} \).

The \( E_{\text{RPS}}(\bar{F}) \) performance value curve has the same behavior as the \( E_{\text{RISK}}(\bar{F}) \) curve. The \( E_{\text{RPS}}(\bar{F}) \) index is a combination of two measures; the training error measure \( E_{\text{RMS}}(\bar{F}) \) and the complexity measure \( \lambda K(\varepsilon) \). The \( E_{\text{RMS}}(\bar{F}) \) measure is a nondecreasing function with respect to \( \varepsilon \) and the penalty term \( \lambda K(\varepsilon) \), knowing \( \lambda^2/N \) is fixed, is a nonincreasing function of \( \varepsilon \) because the number of parameters
Fig. 4.2 Performance curves for $G_{\alpha}$
Fig. 4.3 Performance curves for $G_{r1}$.
Fig. 4.4 Performance curves for $G_{r2}$.
Fig. 4.5 Performance curves for $G_2$.
\(K(\epsilon)\) decreases as the tolerance value \(\epsilon\) increases. Therefore \(ERPS F\) should exhibit a minimum whose location is named \(\epsilon_p\). At small \(\epsilon\) values, \(RMS_{\text{Train}}^T\) is small and the \(ERPS F\) curve follows the curve of the penalty term \((2\lambda^2/N)K(\epsilon)\) and at large \(\epsilon\) values, \(K\) is close to the number of parameters in the parametric subdomain approximation \(\theta\) which is small in comparison to the number of data \(N\). Therefore the penalty term is small and the \(ERPS F\) curve follows the curve of \(ERMS_{\text{Train}}^T\).

The \(E\sigma_{\text{F}}^T\) performance curve approaches the \(E\sigma_{\text{F}}^{\text{Train}}\) curve after some tolerance value named \(\epsilon_3\). For the interval \([\epsilon_3, \epsilon_{\max}]\), \(E\sigma_{\text{F}}^T\) is either equal or close to \(E\sigma_{\text{F}}^{\text{Train}}\) and \(\epsilon_3\) is the lower bound of the region that these two indices are close to each other. Like \(E\sigma_{\text{F}}^{\text{Train}}\), the \(E\sigma_{\text{F}}^T\) index is equal to zero when the tolerance value is equal to \(\epsilon_{\max}\). At \(\epsilon = 0\), unlike \(E\sigma_{\text{F}}^{\text{Train}}\), \(E\sigma_{\text{F}}^T\) is not equal to zero indicating that the deviation among the values of the approximations \(F_C\) at the testing data point is not zero. Although the approximations \(F_C\) have the same values at the training data points when the tolerance value is zero, they have different values at the points remote from the training data points.

We have described the general behavior of the performance curves. In the following sections we intend to investigate the effect of approximation complexity on performance of an MC-HARP approximation for different amplitudes of noise and amounts of data. We studied the variation of performance indices for approximations with low complexity (corresponding to \(\epsilon = \epsilon_{\max}\)), high complexity (corresponding to \(\epsilon = 0\)), and optimal complexity (corresponding to \(\epsilon = \epsilon^*\)). General trends in performance of constructed MC-HARP approximations are extracted to establish a framework for performance analysis of MC-HARP and also a model selection criterion for the optimal complexity.

### 4.6 Performance Behavior of the Simplest MC-HARP Approximation

When \(\epsilon\) is large there is only one subdomain in the partition \(\mathcal{C}\) and all the approximations \(F_C\) and their mean \(F\) are equal to the global parametric approximation \(\theta\) and become the simplest approximations in the families \(F\) and \(F_T\). The parameters \(w\) of the approximation \(\theta\) are computed by minimizing a fitness index such as the sum of the squared errors as follows

\[
\text{minimize } w \quad \frac{1}{N} \sum_{x_i \in \mathcal{S}} \left[ \theta(x_i; w) - (G(x_i) + \lambda \eta_i) \right]^2
\]

The vector of optimal parameters \(w^*\), the solution of the problem (4.17), is a function of the number of data \(N\) and the noise vector \(\lambda \eta\) added to the target values, i.e. \(w^*(N, \lambda \eta)\). Like a general parametric regression problem for a bounded actual function \(G\) and noise vector \(\eta\), the parameters vector \(w^*\) has a limit \(\hat{w}^*\) when the amount of data is large. The limit vector \(\hat{w}^*\) is independent from the amplitude and distribution of the noise vector. (From now on \((\cdot)^*\) is used to indicate the limit of a variable with respect to \(N\)). Regarding this characteristic of a parametric regression, one can write the following relationship

\[
\lim_{N \to \infty} E_{\eta} \left[ \frac{1}{N} \sum_{x_i \in \mathcal{S}} \left[ \theta(x; w^*(N, \lambda \eta)) - G(x) \right]^2 \right]^{\frac{1}{2}} = \left[ \frac{1}{N} \sum_{x_i \in \mathcal{S}} \left[ \theta(x; \hat{w}^*) - G(x) \right]^2 \right]^{\frac{1}{2}} \quad (4.18)
\]
Therefore, by combining Eqns. (4.5) and (4.18), and knowing that $\bar{F}=\theta$, the limit of approximation risk at $\varepsilon=\varepsilon_{\text{max}}$ is computed as follows

$$ERISK_{F}(\varepsilon_{\text{max}}, \lambda) = \left[\frac{1}{N} \sum_{x \in \mathbb{R}} \left[\theta(x; \hat{w}^*) - G(x)\right]^{2}\lambda\right]^{\frac{1}{2}} \tag{4.19}$$

This limit is independent of the amplitude of noise. The approximation risk $ERISK_{F}(\varepsilon_{\text{max}}, \lambda)$ is equal to zero only when the actual mapping $G$ is a member of the family $\Theta$ of the parametric functions $\theta$. As shown in Fig. 4.6 for two functions $G_{r1}$ and $G_{r2}$, the approximation risk converges to a fixed value regardless of the amplitude of noise. For a fixed number of data $N$, the approximation risk $ERISK_{F}(N, \varepsilon_{\text{max}}, \lambda)$ is larger for a large amplitude of noise than the risk for the small amplitude of noise. Also, when the data has a large amplitude of noise, the approximation risk converges more slowly to its limit than when the noise amplitude is small. For noise free data $\lambda=0$, $ERISK_{F}(\varepsilon_{\text{max}}, 0)$ is not equal to zero because $G$ is not in the family $\Theta$ of parametric linear polynomials $\theta$ used for numerical simulations.

Similar to the approximation risk $ERISK_{F}$, there should be a limit for the performance measure $ERMS^{\text{Train}}_{F}$ at $\varepsilon=\varepsilon_{\text{max}}$. However, this limit value, unlike $ERISK_{F}$, is not independent of the amplitude of noise because $ERMS^{\text{Train}}_{F}$ represents a combination of the approximation risk which has a noise-independent limit and the deviation of noise which is a function of the noise amplitude. Using Eqn. (4.7) and knowing that $\bar{F}=\theta$ at $\varepsilon=\varepsilon_{\text{max}}$, one can write

$$ERMS^{\text{Train}}_{F}(\varepsilon_{\text{max}}, \lambda) = \lim_{N \to \infty} E_{\eta} \left[\frac{1}{N} \sum_{x \in \mathbb{R}} \left[\theta(x; \hat{w}^*(N, \lambda\eta)) - (G(x) + \lambda\eta_{k})\right]^{2}\lambda\right]^{\frac{1}{2}} \tag{4.20}$$

Since the vector $\hat{w}^*$ has a limit $\hat{w}^*$ and the random noise variable has a finite variance, the limit defined in Eqn. (4.20) exists and can be written as
The right hand side of Eqn. (4.21) is a function of the noise amplitude $\lambda$. Figure 4.7 shows the variation of $\operatorname{ERMS}_F(N, \varepsilon_{\text{max}}, \lambda)$ with respect to the amount of data and the amplitude of noise. It is evident that $\operatorname{ERMS}_F$ has a limit when the amount of data is large and its limit is directly related to the noise amplitude which means the larger $\lambda$ is, the larger the limit value is. For noise-free data ($\lambda=0$), the limit value $\operatorname{ERMS}_F^{\text{Train}}(\varepsilon_{\text{max}}, 0)$ is only equal to zero if $G$ is in the class $\Theta$. Since the performance index $\operatorname{ERPS}_F$ because close to $\operatorname{ERMS}_F^{\text{Train}}$ when $\varepsilon$ is large, $\operatorname{ERPS}_F(\varepsilon_{\text{max}}, \lambda)$ is equal to $\operatorname{ERMS}_F^{\text{Train}}(\varepsilon_{\text{max}}, \lambda)$.

We have defined $\varepsilon_{\text{max}}$ such that for the tolerance values greater than or equal to $\varepsilon_{\text{max}}$, all the approximations $F_C$ become equal to $\theta$. A mathematical definition for $\varepsilon_{\text{max}}$ can be written as

$$
\varepsilon_{\text{max}} = E_\eta \left[ \max_{x \in \Theta} \left| \theta(x; \omega^*(N, \lambda \eta)) - (G(x_k) + \lambda \eta_k) \right| \right]
$$  \hspace{1cm} (4.22)

Equation (4.22) dictates that $\varepsilon_{\text{max}}$ is a function of the number of data $N$ and the noise amplitude, i.e., $\varepsilon_{\text{max}}(N, \lambda)$. Knowing that the vector $\omega^*$ has a limit for a large value of $N$, one can compute the limit of $\varepsilon_{\text{max}}(N, \lambda)$ using Eqn. (4.22) as follows

$$
\hat{\varepsilon}_{\text{max}}(\lambda) = E_\eta \left[ \max_{x \in \Theta} \left| \theta(x_k^*; \omega^*) - (G(x_k) + \lambda \eta_k) \right| \right]
$$  \hspace{1cm} (4.23)

The limit value $\hat{\varepsilon}_{\text{max}}$ is a function of the amplitude of noise. The boundary values of the other performance curves, $\hat{E}_C^{\text{Train}}$ and $\hat{E}_C^{\text{Test}}$, at $\varepsilon = \varepsilon_{\text{max}}$ are equal to zero and are independent from the values of $N$ and $\lambda$.

The limits with respect to $N$ for $\omega^*$, $\operatorname{ERISK}_F(N, \varepsilon_{\text{max}}, \lambda)$, $\operatorname{ERMS}_F(N, \varepsilon_{\text{max}}, \lambda)$, $\varepsilon_{\text{max}}(N, \lambda)$ and other indices will be defined in the future that are functions of $N$ are finite when the actual function $G$ is bounded and the probability density function for the random noise vector $\eta$ has a bounded support (i.e. the noise variable cannot become infinity). These conditions are generally satisfied for actual data sets. In our numerical
simulation, the random noise vector has a normal distribution whose support is not bounded. However, the limits of the performance indices are not large, as shown in Figs. 4.2 through 4.5, because the variance of the simulated noise variable is finite and we consider a finite number of noisy data sets containing a finite number of data. Therefore there is a small probability that the value of the simulated noise variable becomes large in comparison to the maximum values of the actual functions.

4.7 Performance Behavior of the Most Complex MC-HARP Approximation

When we set the tolerance value \( \varepsilon \) to zero, the approximations \( F_C \) try to fit the noisy data accurately by increasing their complexity. Adding noise to the output of any mapping reduces the smoothness of the mapping throughout its input domain and makes the behavior of the mapping complex in any arbitrarily small region of its domain wherever the noise amplitude is large in comparison to the output of the mapping. Therefore for a perfect fit to the noisy data, the approximations \( F_C \) increase their number of subdomains and reduces the size of their subdomains in order to accurately follow the noisy data. In other words, at \( \varepsilon = 0 \) a constructed mapping \( F_C \) behaves as a local interpolation whose subdomain approximation function \( \theta \) have parameters dominated by noise. The parameters of the approximations \( F_{C(\varepsilon_{\text{max}})} \), like a parametric regression function, are independent of the amplitude of noise when the number of data is large, but the parameters of the local interpolations \( F_{C(0)} \) depend on the noise amplitude for any amount of data. Consequently, the parameters of \( \bar{F}(0) \), that is a linear combination of the local interpolations \( F_C \), are noise dependent. Therefore when \( N \) is large, the performance indices \( \text{ERMS}_{F}^{\text{Train}} \) and \( E_{\sigma_F}^{\text{Test}} \) have limits that are functions of the noise amplitudes, i.e. \( \text{ERMS}_{F}^{\text{Train}}(0, \lambda) \) and \( E_{\sigma_F}^{\text{Test}}(0, \lambda) \).

The parameters of a regression function depend on the noise vector added to the target values. The variation in the values of the parameters with respect to the change in the noise vector decreases as the difference between the number of data \( N \) and the number of parameters \( M \) decreases. The value \( N - M \) is called the number of degrees of freedom and the inverse of the variation in parameters value is referred to as the confidence in parameters values of a regression function. The number of parameters in \( F_{C(\varepsilon_{\text{max}})} \) is a fixed value equal to \( M \) the number of parameters in \( \theta \). Hence by increasing \( N \), we increase the number of degrees of freedom \( N - M \) for the approximation \( F_{C(\varepsilon_{\text{max}})} \) and consequently the variation of the parameters of \( F_{C(\varepsilon_{\text{max}})} \), \( w^*(N, \lambda \eta) \), decreases and the parameters converge to a fixed vector \( \hat{w}^* \). On the other hand, at \( \varepsilon = 0 \), the approximation \( F_{C(0)} \) is piecewise defined by a number of subdomain interpolations \( \theta \) whose number of parameters is close to the number of data points in their corresponding subdomains. Therefore the number of degrees of freedom for \( F_{C(0)} \) is almost equal to zero \( N - K_C = 0 \) (it is zero when the amplitude of noise is large with respect to the actual target values) for any amount of data. So the confidence in parameters values of \( F_{C(0)} \) is low and the variation of their value is directly controlled by the noise vector.

When the subdomain approximation \( \theta \) is linear with respect to its parameters and we set \( \varepsilon \) to be zero for a perfect fit of noisy data, the parameters of \( \theta \) are linear functions of random noise variables. Since \( \bar{F}(0) \) is a linear combination of the functions \( F_{C(0)} \) that are piecewise defined by the subdomain interpolations \( \theta \) whose parameters depend linearly to the amplitude of noise, the values of \( F_{C(0)} \) and \( \bar{F}(0) \) at any point in the input domain are linear functions of the noise amplitude. Consequently, the distance between \( \bar{F}(0) \) and the
actual mapping $G$ represented by the performance index $ERISK_f(N, 0, \lambda)$ becomes linearly dependent on the noise amplitude as the number of data increases. In other words, $ERISK_f(0, \lambda)$ is a linear function of $\lambda$. Figures 4.8 and 4.9 show the variation of $ERISK_f(N, 0, \lambda)$ and $E\sigma^T_{F}(N, 0, \lambda)$, respectively for different numbers of data and amplitudes of noise. There are envelopes for curves showing the variation of these indices with respect to $\lambda$, indicating that $ERISK_f(0, \lambda)$ and $E\sigma^T_{F}(0, \lambda)$ exist and are not equal to zero when data is noisy. By increasing the number of data $N$, the approximation risk $ERISK_f(N, 0, \lambda)$ becomes a linear function of the amplitude of noise demonstrating $ERISK_f(0, \lambda)$ is linear with respect to $\lambda$. It is evident in Figs. 4.8 and 4.9, that the envelopes of the curves in these figures, representing $ERISK_f(0, \lambda)$ and $E\sigma^T_{F}(0, \lambda)$, pass through the origin. For noise-free data, the approximation risk $ERISK_f(N, 0, 0)$ and the deviation measure $E\sigma^T_{F}(N, 0, 0)$ converge to zero as $N$ increases. Indicating that the limit values $ERISK_f(0, 0)$ and $E\sigma^T_{F}(0, 0)$ are equal to zero and for noise-free data, the optimal value of tolerance $\varepsilon^*$ is equal to zero.

The remaining nonzero performance index besides $ERISK_f$ and $E\sigma^T_{F}$ for $\varepsilon=0$ is $ERPS_f$. At $\varepsilon=0$, the number of parameters $K_c$ in the constructed mappings $F_C(0)$ is almost equal to the number of data $N$ for any noise vector $\eta$ added to the target values. Therefore the number of parameters $K$ in the mapping $F(0)$

![Fig. 4.8 Variation of the approximation risk at $\varepsilon=0$](image)

(a) Actual function $G_n$, (b) Actual function $G_n$

![Fig. 4.9 Variation of the standard deviation of approximation at $\varepsilon=0$](image)

(a) Actual function $G_n$, (b) Actual function $G_n$
is almost equal to $N$ for any $\eta$. Using Eqn. (4.14) and knowing that at $\varepsilon=0$, $RMS_{\text{Train}}$ is equal to zero and $K$ can be approximated by $N$, the limit of $ERPS_{\text{Train}}(N, 0, \lambda)$ converges to $\sqrt{2} \lambda$, i.e. $ERPS_{\text{Train}}(0, \lambda) = \sqrt{2} \lambda$. In Figs. 4.2 through 4.5, the boundary values of $ERPS_{\text{Train}}$ curves at $\varepsilon=0$ are equal to $\sqrt{2} \lambda$.

### 4.8 Performance Behavior of the Optimal MC-HARP Approximation

We have shown that the limit of the approximate risk at $\varepsilon=0$, $ERISK_{\text{Train}}(0, \lambda)$, is equal to zero if and only if the data are noiseless. Also, we have shown that at $\varepsilon=\hat{\varepsilon}_{\text{max}}(\lambda)$ the approximation risk converges to zero if and only if the actual function $G$ belongs to the family $\Theta$ of parametric subdomain approximations $\theta$. Therefore the optimal value $\varepsilon^*$ is equal to zero when there is no noise in the data and $\varepsilon^*$ is equal to $\hat{\varepsilon}_{\text{max}}$ when $G$ can be represented exactly by $\theta$. We say an approximation $\bar{F}$ is consistent if its approximation risk $ERISK_{\text{Train}}$ converges to zero as the number of data $N$ increases, i.e. $ERISK_{\text{Train}}=0$. When the data are noisy, $\bar{F}(0)$ is an inconsistent approximation for $G$ because $ERISK_{\text{Train}}(0, \lambda)$ is not equal to zero. When $G$ does not belong to the parametric family $\Theta$, $ERISK_{\text{Train}}(\hat{\varepsilon}_{\text{max}}, \lambda)$ is greater than zero for any value of $\lambda$. Hence, $\bar{F}(\hat{\varepsilon}_{\text{max}})$ is an inconsistent approximation for $G$. Generally, in real-world problems, data are noisy and the actual unknown mapping $G$ is more complex than the assumed parametric family $\Theta$. For these problems, an approximation $\bar{F}$ built by the MC-HARP method is inconsistent when the tolerance value $\varepsilon$ is set to be zero or be equal to its maximum value $\hat{\varepsilon}_{\text{max}}$. Now, one can ask these questions: Is $\bar{F}(\varepsilon)$ inconsistent for all tolerance values in $[0, \hat{\varepsilon}_{\text{max}}]$? Is there an optimal tolerance value $\varepsilon^*$ in $[0, \hat{\varepsilon}_{\text{max}}]$ such that $\bar{F}(\varepsilon^*)$ has the lowest approximation risk in the approximations family $\mathcal{F}$?

We know that when $\bar{F}$ has a small number of parameters like the simple parametric approximation $\bar{F}(\hat{\varepsilon}_{\text{max}})$ or when $\bar{F}$ has a large number of parameters like the local interpolation $\bar{F}(0)$, it has a certain amount of approximation risk that cannot be reduced by increasing the number of data. The approximations $\bar{F}(\varepsilon)$ that are not as complex as $\bar{F}(0)$ and are not as simple as $\bar{F}(\hat{\varepsilon}_{\text{max}})$ have smaller approximation risk. By decreasing the tolerance value $\varepsilon$ starting from $\hat{\varepsilon}_{\text{max}}$, we increase the complexity and flexibility of the approximation $\bar{F}(\varepsilon)$ and consequently we reduce the approximation risk to be less than $ERISK_{\text{Train}}(\hat{\varepsilon}_{\text{max}})$.

On the other hand, if we increase the complexity and flexibility of $\bar{F}(\varepsilon)$ too much by setting $\varepsilon$ to be small, then the output of the constructed approximation $\bar{F}(\varepsilon)$ is dominated by noise and $ERISK_{\text{Train}}$ increases as $\varepsilon$ becomes closer to zero. Therefore there is an approximation with optimal complexity corresponding to an optimal tolerance value $\varepsilon^*$. For a given data, $\bar{F}(\varepsilon^*)$ has the lowest approximation risk in the family $\mathcal{F}$. The approximations $\bar{F}(\varepsilon)$ with $\varepsilon > \varepsilon^*$ are not complex enough and do not have optimal flexibility and the approximations $\bar{F}(\varepsilon)$ with $\varepsilon < \varepsilon^*$ are not simple enough and have more than the optimal number of parameters.

Figure 4.10 shows the variation of the approximation risk with respect to the tolerance value for different numbers of data and amplitudes of noise. The existence of the optimal tolerance value is evident. The optimal tolerance value $\varepsilon^*$ is small when the noise amplitude is small and $\varepsilon^*$ increases as the noise amplitude increases. For a fixed amplitude of noise, the value of $\varepsilon^*$ is a function of the number of data $N$. The dependency of $\varepsilon^*$ on $N$ indicates that the optimal tolerance value for one data set may not be optimal for another data set. However, since $\varepsilon^*(N, \lambda)$ has a limit value called $\hat{\varepsilon}^*(\lambda)$, when the number of data points is large enough, $\varepsilon^*(N, \lambda)$ is close to $\hat{\varepsilon}^*(\lambda)$ and does not change significantly with respect to $N$. 

97
Fig. 4.10 Variation of the approximation risk with respect to the tolerance value for different numbers of data and amplitudes of noise
To see how the complexity of $F(\varepsilon)$ changes with respect to the quality and quantity of data, in Fig. 4.11 we show the variation in the logarithm of the number of subdomains $E\bar{s}(N, \varepsilon, \lambda)$ for an approximation $\bar{F}$ with respect to the tolerance value for different number of data and amplitudes of noise. $E\bar{s}$ is the expected value with respect to the noise vector $\eta$ of $\bar{s}$ which is the expected value of the number of subdomains $s_C$ with respect to C, i.e. $E\bar{s}=E_{\eta}[s_C]$ and $s=E_{\eta}[s_C]$. It can be observed that the number of subdomains for $\bar{F}$ decreases as the tolerance value increases indicating that $E\bar{s}(N, \varepsilon, \lambda)$ is a decreasing function of $\varepsilon$ for a fixed value of $N$ and $\lambda$. Also, the number of subdomains $E\bar{s}$ increases as the amplitude of noise increases showing that $E\bar{s}(N, \varepsilon, \lambda)$ is an increasing function of $\lambda$ for a fixed value of $N$ and $\varepsilon$. The $E\bar{s}$ value has the tendency to increase as the number of data $N$ increases. The value of $E\bar{s}$ is equal to one when $\varepsilon$ reaches its maximum value $\varepsilon_{\max}$. At $\varepsilon=0$, the number of subdomains in $\bar{F}$, $E\bar{s}(N, 0, \lambda)$, is almost independent of the amplitude of noise. The reason for this characteristic is as follows: at $\varepsilon=0$, each constructed function $F_C$ is a local interpolation whose subdomains contain the same number of data points as the number of parameters in their corresponding subdomain functions $\theta$. These subdomain interpolations $\theta$ can fit data with any target values and consequently data with any amplitude of noise. Therefore for a fixed number of data, a random partition $C$ developed to build a local interpolation $F_C$ can be used to perfectly fit data with any amplitude of noise just
by adjusting the parameters of subdomain interpolations \( \theta \) without changing the size and number of subdomains in \( \mathbb{C} \). Hence, \( E^\mathbb{C}(N, 0, \lambda) \) is almost independent from \( \lambda \). Figure 4.11 shows that when the amplitude of noise is small and the number of data is large, the logarithm of the number of subdomains \( E^\mathbb{C} \) decreases with a slow rate for a range of \( \varepsilon \) values and almost a saddle region is developed along the curves of \( \log E^\mathbb{C} \) with respect to \( \varepsilon \). By increasing the amplitude of noise or decreasing the number of data, the size of this saddle region is reduced. In our simulations, the lower bound of this saddle region is very close to the optimal tolerance value \( \varepsilon^* \). This characteristic may be helpful to select the optimal tolerance value when the amplitude of noise is small and the number of data is large.

Figure 4.12 shows the variation of the number of subdomains for the optimal approximation \( \overline{F}(\varepsilon^*) \) with respect to the number of data and amplitude of noise. Figure 4.13 compares the rate of increasing for the number of subdomains with the rate of increasing data. The number of subdomains of the optimal approximation \( \overline{F}(\varepsilon^*) \) is a function of the number of data \( N \) and the amplitude of noise \( \lambda \), i.e. \( E^\mathbb{C}(N, \varepsilon^*, \lambda) \).

For a fixed amplitude of noise, by increasing the number of data \( N \), \( E^\mathbb{C} \) increases but sufficiently slowly such
that \( E_\mathbb{S}(N, \varepsilon^*, \lambda)/N \) converges to zero with respect to \( N \). The slower rate of increasing for \( E_\mathbb{S}(N, \varepsilon^*, \lambda) \) in comparison to the rate of increasing data makes the average number of data points in each subdomain, represented by \( N/E_\mathbb{S} \), increases as the number of data increases. Therefore as \( N \) increases, the confidence in the parameters of subdomain approximations \( \theta \), which is directly related to \( N/E_\mathbb{S} \), increases. For a fixed number of data, \( E_\mathbb{S}(N, \varepsilon^*, \lambda) \) decreases as the amplitude of noise increases. Indicating that when the amount of data is fixed, for construction local approximations with low risk the subdomains should be expanded with respect to the amount of noise. For data with a large amount of noise, the size of subdomains is greater than for data with a small amount of noise. The MC-HARP tries to expand subdomains when data does not have good quality in order to include more data in subdomains for building subdomain approximation \( \theta \) that capture the main local features of the data that have not been destroyed by noise. The smaller the amplitude of noise is, the more evident the local features of data are. Hence for noisy data, subdomains should be small enough to capture the evident local features in data but not too small.

Therefore the approximation algorithm builds optimal approximations \( \bar{F}(\varepsilon^*) \) that converge to the actual mapping \( G \) by shrinking subdomains in a controlled manner. The size of subdomains in the optimal partitions is large enough to prevent subdomain approximations \( \theta \) from following the noise. The number of subdomains increases with a rate slower than the growth rate of data supplying enough data in each subdomain to construct a confident subdomain approximation. The optimal approximations built by the MC-HARP method approach the actual mapping with confidently estimated parameters and low risk of noise dominance.

4.9 Approximation Risk of an MC-HARP Approximation:
Optimal Complexity and Consistency

We have shown that the approximation risk is a function of the complexity of the constructed approximation and there exists an approximation with optimal complexity that has the lowest approximation risk for a given data set. Also, we have shown that the complexity of an approximation built by the MC-HARP method is directly represented by the average number of subdomains in the sample of random partitions when the subdomain approximation function is fixed. Finally, we have shown that the number of subdomains is a decreasing function of the tolerance value. Therefore we can conclude that since the approximation risk is a function of complexity and since complexity is directly related to a decreasing function of the tolerance value, then the approximation risk is a function of the tolerance value and there exists an optimal tolerance value corresponding to an approximation with optimal complexity and the lowest approximation risk. Figure 4.14 shows how the value of the optimal tolerance varies with respect to the number of data and amplitude of noise. It is evident that \( \varepsilon^*(N, \lambda) \) has a limit value called \( \hat{\varepsilon}^*(\lambda) \) for large data sets.

The approximation risk for \( \bar{F}(\hat{\varepsilon}^*) \) decreases as the number of data increases, as shown in Fig. 4.10, indicating that \( \text{ERISK}_F(\hat{\varepsilon}^*, \lambda) \) is equal to zero. Therefore \( \bar{F}(\hat{\varepsilon}^*) \) is a consistent approximation for the actual mapping \( G \). When data is noiseless, \( \hat{\varepsilon}^* \) is equal to zero and when the actual function \( G \) belongs to the parametric family \( \Theta \) of the subdomain approximation \( \theta \), \( \hat{\varepsilon}^* \) is equal to \( \hat{\varepsilon}_{\text{max}} \). When \( G \) does not belong to \( \Theta \) and data is noisy, \( \hat{\varepsilon}^* \) cannot be equal to zero or \( \varepsilon_{\text{max}} \), because \( \bar{F}(0) \) and \( \bar{F}(\varepsilon_{\text{max}}) \) are inconsistent and \( \bar{F}(\hat{\varepsilon}^*) \) is a consistent approximation. Through numerical simulations, we have observed that for a fixed subdomain approxi-
mation $\theta$, amplitude of noise $\lambda$ and number of random partitions $p$, there is a unique tolerance value whose corresponding approximation is consistent. This tolerance value is $\hat{\epsilon}$. The approximation $F(\epsilon)$ with $\epsilon \neq \hat{\epsilon}$ are inconsistent approximation for $G$ and their approximation risk does not converge to zero as the number of data increases.

The consistency of $F(\hat{\epsilon})$ should not be interpreted that the approximation risk of $F(\hat{\epsilon})$ can become equal to zero when the number of data is large. The consistency of $F(\hat{\epsilon})$ means that the approximation risk of $F(\hat{\epsilon})$ can approach zero as close as we want if the number of data is large enough. When $G$ does not belong to $\Theta$, there is zero probability that an actual mapping is exactly represented by a local approximation $F_C$ because of the random nature of partitions $C$. Therefore the approximation risk of the constructed mappings $F_C$ built by the HARP method and $\tilde{F}$ built by the MC-HARP method cannot become equal to zero if the actual function does not belong to $\Theta$.

The MC-HARP method tries to approximate an actual mapping by constructing a sample of mappings that do not have the same structure as the actual mapping. This characteristic of the MC-HARP strategy makes it be fair and indiscriminate with respect to all multivariate mappings. The MC-HARP method is a universal approximation method whose performance does not depend upon the structure of the actual mapping and has the same performance no matter the actual mapping is homogeneous, piecewise defined, harmonic, exponential, additive, and so on. Of course one can use a priori knowledge about the structure of the actual mapping to improve the performance of the MC-HARP method in different ways such as defining the parametric family $\Theta$ to be close to the class of the actual mapping and controlling the partitioning scheme to follow faster the complexity of the actual mapping.

The behavior of the approximation risk of $\tilde{F}$ with respect to the amplitude of noise and tolerance value for large number of data, $\text{ERISK}_F(\epsilon, \lambda)$, is schematically shown in Fig. 4.15. This figure summarizes all the characteristics of the approximation risk explained previously. At $\epsilon = 0$, $\text{ERISK}_F$ is an increasing function of the noise amplitude and specifically when the subdomain approximations $\theta$ are linear-in-parameters, $\text{ERISK}_F(0, \lambda)$ is a linear function of $\lambda$. When $\epsilon$ is large ($\epsilon = \epsilon_{\text{max}}$), the approximation risk is constant for any
amplitude of noise. For a fixed noise amplitude and a large number of data, by increasing the tolerance value \( \varepsilon \), the approximation risk decreases and reaches a minimum value close to zero at an optimal tolerance value and then increases and reaches a fixed value. As the amplitude of noise increases, the value of the optimal tolerance increases and takes distance from zero. The shape of the \( \text{ERISK}_F(\varepsilon, \lambda) \) surface indicates that for moderately large amplitude of noise, the simple parametric approximation \( F(\varepsilon^*) \) has smaller approximation risk than the complex local interpolation \( F(0) \). This characteristic suggests that when we do not know the optimal tolerance value, it is safer to build simple approximations than the complex ones.

The variation of the approximation risk with respect to the approximation complexity and quality of data demonstrates the importance of selecting the tolerance value in order to build an MC-HARP approximation with acceptable risk using a given data set.

4.10 Conclusions

We have shown that the tolerance value \( \varepsilon \) for the termination criterion for the subdomain training process of MC-HARP controls the distribution, size, and number of subdomains in a HARP partition. Furthermore the number of parameters in an MC-HARP approximation is also controlled by \( \varepsilon \). The tolerance for the approximation error controls the HARP approximation-error-driven partitioning scheme and its corresponding local approximation. The tolerance value \( \varepsilon \) is the only complexity-controlling parameter for an MC-HARP approximation \( \bar{F} \). The structural details of \( \bar{F} \), including the distribution, size, and number of subdomains, are automatically determined by the HARP algorithm. A HARP partition \( \mathcal{C} \), HARP approximation \( F_\mathcal{C} \), and MC-HARP approximation \( \bar{F} \) are functions of \( \varepsilon \).

The model selection problem for an MC-HARP approximation is defined to be: the selection of the optimal tolerance \( \varepsilon^* \) such that its corresponding MC-HARP approximation \( \bar{F}(\varepsilon^*) \) has the minimum approximation risk in the family of approximations built by MC-HARP with different complexities.

We have described techniques needed only the training data to estimate the approximation risk. There are two main classes of model selection techniques in the literature: (1) techniques based on resampling data;
namely, the cross-validation and bootstrap methods and (2) techniques that penalize the training error according to the complexity of the constructed approximation; namely, the root-predicted-squared error and the minimum-description-length criteria.

We have studied the characteristics of the MC-HARP method for noisy data through numerical simulations. The general behaviors of performance indices for the quantity-quality domain are determined. The performance of an MC-HARP approximation with low complexity (corresponding to \( \varepsilon = \varepsilon_{max} \)), high complexity (corresponding to \( \varepsilon = 0 \)), and optimal complexity (corresponding to \( \varepsilon = \hat{\varepsilon}^* \)) are investigated. General trends in performance of constructed MC-HARP approximations are extracted to establish a framework for performance analysis of MC-HARP and also a model selection criterion for the optimal complexity.

It is shown: (1) the approximation risk is a function of the complexity of the constructed approximation and there exists an approximation with optimal complexity that has the lowest approximation risk for a given data set, (2) the complexity of an approximation built by the MC-HARP method is directly represented by the average number of subdomains in the sample of random partitions when the subdomain approximation function is fixed, and (3) the number of subdomains is a decreasing function of the tolerance value. Therefore the approximation risk is a function of the tolerance value and there exists an optimal tolerance value corresponding to an approximation with the optimal complexity and lowest approximation risk.

We have shown that the optimal tolerance \( \varepsilon^* \) is equal to zero for noise-free data and is greater than zero for noisy data. Furthermore the simplest MC-HARP approximation corresponding to \( \varepsilon = \varepsilon_{max} \) is optimal only when the actual mapping belongs to the family of the parametric functions that have the functional form of the selected subdomain approximation for MC-HARP. The value of the optimal tolerance is a function of the amplitude of noise added to the data. For a noisy data set, the MC-HARP approximations with \( \varepsilon \neq \hat{\varepsilon}^* \) are inconsistent approximation and their approximation risk does not converge to zero as the number of data increases. The inconsistent approximations corresponding to \( \varepsilon = \varepsilon_{max} \) and \( \varepsilon = 0 \) behave like a global approximation and a local interpolation, respectively. HARP and MC-HARP approximations are consistent only for the unique value of the optimal tolerance.
CHAPTER FIVE
A New Philosophy for Performance Estimation
of Data-based Approximate Mappings

"Non sunt multiplicanda entia praeter necessitatem."
(Entities are not to be multiplied beyond necessity.)
William of Ockham

"The by-product is sometimes more valuable than the product."
Havelock Ellis

5.1 Current Practice in Nonparametric Approximation

The most reliable criterion for selecting an approximation from a family of approximations or testing the performance of a constructed approximation for a given data set is the prediction error computed for a test set that covers the input domain and includes all main features of the actual mapping. Unfortunately, in practice we usually do not have enough data to organize a reliable test set. Researchers have got around this problem by developing techniques for estimating the prediction error from the given data set. There are two main classes of model selection techniques, as discussed in Section 4.2: (1) techniques based on resampling data; namely, the cross-validation and bootstrap methods and (2) techniques that penalize the training error according to the complexity of the constructed approximation; namely, the root-predicted-squared error and the minimum-description-length criteria. Each model selection technique is well defined for a specific category of mathematical models. The larger this category is, the more general the technique is.

The resampling techniques are computationally expensive, especially when the constructed approximation is complex and the penalizing techniques need prior knowledge of the distribution of random noise added to data. In addition to these shortcomings, the estimate of the prediction error computed by these techniques is biased, although the bias is small when the data set is large enough. Hence these techniques can reliably select the best approximation only when the data set is large. For data sets of moderate size, they may select an overparameterized or oversimplified approximation. It seems that in spite of using these model selection techniques, we still need a large data set to reliably select the best mathematical model for a given data set. The main advantage of using these techniques is that we can always compute an estimate, albeit biased, for the prediction error no matter what the size of the data set is and then use this estimate as a criterion to select an approximation in a family of approximations. Although the selected model may not be the best approximation, it is the best for the chosen criterion.
Different model selection criteria may lead to the selection of different approximations as the best of a family of approximations. When the data set is large, these criteria will almost surely select the same approximation, that is, the approximation with the lowest prediction error. The agreement among different model selection criteria for large data sets can be used as a qualitative index of the confidence in the constructed mathematical model. When for a given data set, different criteria agree on the selected mathematical model, we have better confidence in the constructed model than when different criteria lead to different models. The confirmation of an approximation by different criteria is a necessary condition for the adequacy of data and suitability of the approximation.

Since all model selection techniques use statistical methodologies to compute an estimate of the prediction error and the problem of fitting data has a stochastic nature, all model selection criteria are stochastic quantities. Consequently, there is a low probability that these stochastic estimates of the prediction error would be equivalent when the data set is inadequate. On the other hand, a number of criteria might agree on a model even when the amount of data is actually insufficient. Thus, one should be careful about interpreting estimated values of the prediction error computed by different model selection techniques and the variation of these estimates with respect to one another.

The main disadvantage of currently available model selection techniques is that one can construct an approximation representing the given data for any amount of data with any amplitude of noise. There is no general criterion for checking the sufficiency of data and preventing the construction of nonparametric mathematical models for data sets that do not adequately represent the essential features of the actual mapping. Also, there is no robust technique for computing reliable estimates for the prediction error and the confidence of a constructed approximation when the data set is not large. Finally, there is no solid mathematical criterion for determining the quantity and quality of a data set. In other words, there are no accurate and objective definitions for terms like small data set, large data set, small amount of noise, and large amount of noise.

Unlike the parametric approximation, the nonparametric approximation does not have a well defined framework for computing the confidence interval for an output of the constructed approximation and for verifying the adequacy of available data. The need for objective measures of the confidence in an approximation and the sufficiency of data is evident. These measures can be used to define when a nonparametric data-fitting problem is well-conditioned and can have a useful and meaningful solution. We believe that, like any mathematical problem, a data-fitting problem may be ill-conditioned. An ill-conditioned data-fitting problem is one where the available data are inadequate to completely represent the main features of the actual mapping that distinguish it from other mappings. One should have little confidence in the solution of an ill-conditioned problem, since it does not reliably capture the main features of the actual mapping. An ill-conditioned mapping approximation problem has many useless solutions and its data may be represented by any of several mappings. There should be a framework for detecting ill-conditioned problems and warning the experimenter about the inadequacy of available data and hence the dubious quality of the mathematical model constructed from the given data. Conversely, the proposed framework should be able to distinguish a well-conditioned problem, that is, one having enough data to build an approximation with good confidence.
The MC-HARP method approaches an actual mapping by building a sample of approximated mappings. All other approximation methods try to find only one mathematical model. Also the approximations built by the MC-HARP method for the optimal tolerance value are consistent. Building a sample of consistent approximations is the main characteristic of the philosophy behind the MC-HARP method that is responsible for the robustness and feasibility of the MC-HARP method for dealing with noisy data. Furthermore, this characteristic enables the MC-HARP method to have its own model selection criterion compatible with its philosophy, to define quantitative measures of the confidence in approximation and the adequacy of data, and finally to establish a framework for classifying data-fitting problems. In the rest of this chapter we explain these capabilities of the MC-HARP method.

5.2 New Criterion for Selecting the Best Approximation

The MC-HARP method approximates an actual mapping $G$ by building a sample of local approximations $F_C$. Because of the random nature of partitions $C$, the constructed approximations $F_C$ cannot be exactly equal to the actual mapping, but they can be arbitrarily close to $G$ if the amount of data is sufficient. For the optimal tolerance value $\hat{e}^*$, by increasing the number of data $N$, a local approximation $F_C(\hat{e}^*)$ converges to the actual mapping $G$. The approximation risk $ERISK_{F_C(\hat{e}^*)}$ which measures the distance between $F_C(\hat{e}^*)$ and $G$, converges to zero as the number of data increases. The consistency of approximation $F_C(\hat{e}^*)$ dictates the consistency of their sample mean average $\overline{F}(\hat{e}^*)$. In other words, the convergence of the constructed mappings $F_C(\hat{e}^*)$ toward the actual mapping dictate the convergence of their mean $\overline{F}(\hat{e}^*)$ and vice versa. Furthermore, for the optimal tolerance value, the distance between $\overline{F}(\hat{e}^*)$ and the actual mapping $G$, the approximation risk $ERISK_{\overline{F}(\hat{e}^*)}$, converges to zero as the number of data increases. The following hypothesis is proposed:

**Hypothesis:** If, by increasing the amount of data, the sample of constructed mappings $F_C(\hat{e}^*)$ converge toward the actual mapping $G$, then their deviation measure $E\sigma_{F_C(\hat{e}^*)}$ converges to zero and vice versa.

**Reasoning:** The convergence of the sample of approximations $F_C(\hat{e}^*)$ and its mean toward the actual mapping suggests that the sample deviation $E\sigma_{F_C(\hat{e}^*)}$ should converge to zero. The deviation measure $E\sigma_{F_C(\hat{e}^*)}$ represents the mean distance between each approximation $F_C(\hat{e}^*)$ and the sample mean $\overline{F}(\hat{e}^*)$. The mean distance $E\sigma_{F_C(\hat{e}^*)}$ is smaller than the sum of the distance between an approximation $F_C(\hat{e}^*)$ and the actual mapping $G$ and the distance between the mean approximation $\overline{F}(\hat{e}^*)$ and $G$ (by the triangle inequality). Since the distances between $F_C(\hat{e}^*)$ and $G$ and between $\overline{F}(\hat{e}^*)$ and $G$ converge to zero, the mean distance $E\sigma_{F_C(\hat{e}^*)}$ must converge to zero.

Conversely, the convergence of the deviation measure $E\sigma_{F_C(\hat{e}^*)}$ to zero, indicates that the sample of the constructed mappings $F_C(\hat{e}^*)$ converge toward a unique mapping. Uniqueness means that the sample of mapping $F_C(\hat{e}^*)$ cannot converge to different mappings because if they did, their deviation measure would not converge to zero. Since the constructed mappings $F_C(\hat{e}^*)$ are built based on the random partitions of the input domain and the construction processes for these mappings are independent of one another, therefore, the probability that they are equal to one another is zero. The main, common thread between these constructed mappings is the given training data set. The only way that all of these different mappings $F_C(\hat{e}^*)$ could con-
verge to a unique mapping is that they represent the actual mapping $G$ whose features are contained in the given data and can be shared by these constructed mappings. In other words, the dissimilarity among the mapping constructed by the MC-HARP method and the convergence of their deviation measure to zero are two factors that dictate the approximation risk $ERISK_{F_C(\hat{e}^*)}$ for the sample of constructed mappings $F_C(\hat{e}^*)$ and consequently, the approximation risk $ERISK_{F(\hat{e}^*)}$ for their mean $\overline{F}(\hat{e}^*)$ converge to zero as the amount of data increases. ■

For the optimal tolerance value $\hat{e}^*$, both the approximation risk $ERISK_{F(\hat{e}^*)}$ and the approximation deviation $E\sigma_{F_C(\hat{e}^*)}$ converge to zero as the amount of data increases. This characteristic of the solution sample developed by the MC-HARP method can be used to select the optimal tolerance value. The performance measure $E\sigma_F$, unlike the measure $ERISK_F$, can be computed without knowing the actual mapping $G$ and, more importantly, without knowing target values at any point in the input domain. Of course we need target values for training data points to build local approximations $F_C$ but we can compute the approximation deviation $\sigma_F$ at any point in the input domain without knowing the noisy or actual target value corresponding to that point.

We propose to use $E\sigma_F$ as a model selection criterion. At the optimal tolerance value, $E\sigma_F$ converges to zero as the number of data increases;

$$\varepsilon \in [0, \varepsilon_{\max}) : \lim_{N \to \infty} E\sigma_{F_C(\varepsilon)} = 0 \iff \varepsilon = \hat{\varepsilon}^*$$ (5.1)

To present how this characteristic can be applied for selecting the optimal tolerance, we show in Figs. 5.1 and 5.2 the evolution of performance curves $E\sigma_F^{Train}$, $E\sigma_F^{Test}$, and $ERMS_F^{Train}$, for the functions $G_{r1}$ and $G_{r2}$ with respect to the quantity of data represented by the number of data $N$ and the quality of data represented by the noise amplitude $\lambda$. It is evident that, for a fixed amplitude of noise, by increasing the number of data, a local minimum is generated on the performance curves $E\sigma_F^{Train}$ and $E\sigma_F^{Test}$ at the optimal tolerance value $\hat{\varepsilon}^*$. We expect the development of a local minimum on both performance curves $E\sigma_F^{Train}$ and $E\sigma_F^{Test}$ because at the optimal tolerance value these performance measures should converge to zero as the number of data increases and the only way for positive curves like $E\sigma_F^{Train}$ and $E\sigma_F^{Test}$ to converge to zero at one point in their domains is to generate a local minimum around that point. The convergence of both performance measures $E\sigma_F^{Train}$ and $E\sigma_F^{Test}$ indicates that the approximations $F_C$ built by the MC-HARP method for the optimal tolerance value have the same approximation deviation all over the input domain represented by the training and testing data points. This characteristic indicates that approximations $F_C(\hat{\varepsilon}^*)$ and their mean $\overline{F}(\hat{\varepsilon}^*)$ converge to the actual mapping pointwise almost everywhere.

Unlike the current model selection techniques that use an estimate of the prediction error as their model selection criterion and minimize it to select the best approximation, we have proposed our model selection criterion as a measure of deviation among a sample of local approximations and we select the best approximation to be the mean of the approximation sample whose deviation measure converges to zero as the amount of data increases. Our proposed model selection philosophy is based on the minimization of the deviation measure $\sigma_F$ in the limit with respect to the amount of data over the entire input domain. The convergence in the limit is the necessary condition for the sample of approximations $F_C(\hat{\varepsilon}^*)$ to be consistent. The minimiza-
Fig. 5.1 Evolution of performance curves and quality-quantity map for the function $G_{r1}$
Fig. 5.2 Evolution of performance curves and quality-quantity map for the function $G_2$. 

$N^* (\lambda) \quad \lambda = \lambda^*$ 

Amplitude of noise, $\lambda$ 

Number of data, $N$ 

$E_{\text{RMSE}}^{\text{Train}}, E_{\text{F}}^{\text{Test}}, E_{\text{F}}^{\text{Train}}, e^*$
tion in the limit and consideration of the entire input domain are the two main characteristics of the proposed model selection technique that distinguish it from current techniques which all use a straight minimization over the set of given data points or a subset of it.

The convergence in the limit is objectively represented by the development of a local minimum in both performance curves $\sigma_F^{\text{Train}}$ and $\sigma_F^{\text{Test}}$. Unlike the current model selection criteria that always select an approximation as the best one regardless the amount of available data, the proposed criterion only selects the best approximation when the data set is large enough to develop a distinguished local minimum. From now on we say the amount of data is adequate or the number of data is large when the local minimum is clearly developed on the performance curves $\sigma_F^{\text{Train}}$ and $\sigma_F^{\text{Test}}$. Thus, we have defined an objective measure for the adequacy of data. The proposed criterion overcomes the gap between current model selection criteria and the adequacy of data.

A technique to check the convergence of the deviation measure $\sigma_F$ for the entire input domain is to compute the deviation measure for a test set in addition to the training set. The deviation measure for a set of points in the input domain is the mean average of the standard deviation of the sample of predicted outputs computed for these points. The set of test data points should be disjoint from the set of training data points and uniformly cover the input domain with high resolution. The test data points may be equally distant from one another for low dimensional problems. For high dimensional problems, a set of uniformly random distributed points is a reasonable choice of test set. The number of test points depends on the amount of accessible computational power. The larger is the test set, the more reliably the deviation measure $\sigma_F^{\text{Test}}$, computed for the test set represents the deviation measure of the constructed approximation over the entire input domain. We recommend that the number of test points be equal to or larger than the number of training data points.

The current model selection techniques calculate a performance estimate, i.e. the estimates $CV_f$, $BS_B$, or $RPS$ of the prediction error, only on the set of given data points. However, our proposed deviation measure can be computed not only for the given data points but also for other points in the input domain because for computing the proposed measure at any point in the input domain, one does not need to know the noisy or actual target value at that point. The beauty of the MC-HARP method is that it can generate two performance curves, $\sigma_F^{\text{Train}}$ and $\sigma_F^{\text{Test}}$, to offer more insight into the performance of its constructed approximations over the entire input domain. These curves contain an extensive amount of information. We have already stated how one can use them to select the best approximation in the family $F$. We will explain in the following sections how these performance curves can establish a framework for classifying data-fitting problems based on the objective measures of the adequacy of data and the confidence in approximation. In the following sections, we first define the concepts of confidence and accuracy in approximation, we then define several critical points on the performance curves $\sigma_F^{\text{Train}}$ and $\sigma_F^{\text{Test}}$, we next introduce our classification framework for database-based mathematical modeling problems, and we finally apply our proposed framework to build a mathematical model for a set of real data.
5.3 Performance Estimation of Approximate Mappings: Approximation Deviation and Confidence in Approximation

We define the measure of the deviation in approximation at a point in the input domain, for an approximation \( \bar{F} \) built by the MC-HARP method, to be the standard deviation of outputs predicted by the sample of approximations \( F_C \), whose mean average is \( \bar{F} \), for that point. The integral of pointwise computed deviations over the input domain divided by the volume of the input domain is the measure of the deviation in approximation for the sample of approximation \( F_C \) corresponding to \( \bar{F} \) that we simply refer to as the approximation deviation measure of \( \bar{F} \); i.e.,

\[
\text{approximation deviation measure of } \bar{F} = \frac{1}{V} \int_{\Omega} \sigma_F(x) \, dx
\]  

(5.2)

The approximation deviation measure for a point or a set of points represents the precision in approximation for that point or set. The smaller is the approximation deviation measure, the higher is the precision in approximation. An approximation for the approximation deviation measure of \( \bar{F} \) can be computed by averaging the pointwise defined approximation deviation measures over a finite discrete subset of the input domain namely the training and/or test sets.

The approximation deviation measures for different tolerance values for the training set and the test set are represented by the performance curves \( \sigma_F^{\text{Train}} \) and \( \sigma_F^{\text{Test}} \), respectively. When the test and/or training sets cover the entire input domain with good resolution and the deviation measures \( \sigma_F^{\text{Train}} \) and \( \sigma_F^{\text{Test}} \) are almost equal, we say the approximation deviation of \( \bar{F} \) is measurable and its value is reliably equal to \( \sigma_F^{\text{Train}} \) or \( \sigma_F^{\text{Test}} \), otherwise the approximation deviation of \( \bar{F} \) is not measurable; i.e.,

\[
\sigma_F^{\text{Test}} = \sigma_F^{\text{Train}} \iff \text{approximation deviation measure of } \bar{F} \text{ is measurable}
\]  

(5.3)

For example, at \( \varepsilon = 0 \) the deviation measure \( \sigma_F^{\text{Train}} \) is equal to zero but \( \sigma_F^{\text{Test}} \) is greater than zero, therefore the approximation deviation measure of \( \bar{F}(0) \) is not measurable. On the other hand, at \( \varepsilon = \varepsilon_{\text{max}} \) both measures \( \sigma_F^{\text{Train}} \) and \( \sigma_F^{\text{Test}} \) are equal to zero, hence the approximation deviation measure of \( \bar{F}(\varepsilon_{\text{max}}) \) is equal to zero. At \( \varepsilon = 0 \), the constructed approximation \( \bar{F}(0) \) has high precision for training data points but lower precision for test points and at \( \varepsilon = \varepsilon_{\text{max}} \) the constructed parametric approximation \( \bar{F}(\varepsilon_{\text{max}}) \) has high precision over the entire input domain.

The deviation among a sample of approximations built for a given data set is also used by statistical methods like cross-validation and bootstrap techniques as a performance measure, but based on a philosophy different from the MC-HARP philosophy of approximation. These methods build their sample of approximations using a fixed parametric or nonparametric approximator with a sample of training sets taken from the given data set. The sampling of training sets from the given data can be based on different schemes including the random selection of a subset in the cross-validation technique and the sampling with replacement in the bootstrap technique. In the MC-HARP method we do not partition the data set, we partition the input domain.
These two different philosophies for estimating the performance of a constructed mapping are shown schematically in Fig. 5.3.

5.3.1 Sampling-based Philosophy for Performance Estimation of Approximate Mappings

The sampling-based philosophy tries to measure the sensitivity of the constructed approximation with respect to the available data. This class of performance estimators deals with a mapping approximation problem like the classical statistical problem of estimating the mean of a random variable from a sample of its population. The constructed approximation is the estimated mean (regression function) and the given data is the sample of observations. The main idea behind the sampling-based performance estimators is to measure the performance sensitivity of the constructed approximation (estimated mean) when a randomly chosen portion of the given data (sample of observations) is not presented to the approximator (mean estimator). The deviation in approximation (standard error) is measured over the training set or a subset of it (hold out portions of data) and is used as a statistical measure of precision or accuracy of the constructed approximation. These techniques are widely used to build parametric and nonparametric mathematical models. The sensitivity with respect to holding out some data is a reasonable criterion for preventing the construction of overparameterized models (model selection technique) and for selecting the most significant independent variables. Also these resampling techniques have been extended to estimate standard errors, confidence intervals, and other measures of statistical accuracy (Efron and Tibshirani 1986). We discussed the main shortcomings of these techniques in Section 5.1.

Remark. The combination of the MC-HARP method and a sampling-based performance estimator is a reasonable procedure for building data-based mathematical models. As shown in Fig. 5.3, the MC-HARP

Fig. 5.3 Performance estimation techniques
approximator can be used as the approximator for estimating the performance by sampling-based techniques. The best approximation in the family \( \mathcal{F} \) can be selected using a sampling-based criterion such as cross-validation or bootstrap. Although the MC-HARP method has its own performance estimator and model selector, we recommend using other performance estimators, whenever possible, in order to evaluate the performance measure computed by the MC-HARP method and to double check the optimal tolerance value selected by the MC-HARP method with other model selection criteria. The computational efficiency is a significant factor for selecting a performance estimator for the MC-HARP method. For example, using resampling techniques including bootstrap and cross-validation methods, with the MC-HARP approximator is computationally intensive because the MC-HARP training process must be repeated for all resampled training sets. The MC-HARP philosophy can also offer a model selection technique that is naturally compatible with the MC-HARP approximation method and is not separate from the computational process of building an approximation by the MC-HARP method. Using the MC-HARP performance estimator with the MC-HARP approximator is more computationally efficient than using the MC-HARP approximator with a cross-validation or bootstrap type of model selection techniques. This shortcoming of current performance estimators and their other shortcomings, that have been or will be explained, begged the development of MC-HARP's own performance estimation philosophy.

5.3.2 MC-HARP Philosophy for Performance Estimation of Approximate Mappings

The MC-HARP philosophy for estimating the performance of its constructed approximation at a point in the input domain is to compute the deviation among outputs predicted by a sample of local parametric approximations whose supports contain that point and are built using data points near that point. The predicted output of a local approximation \( F_c \), built by the HARP method, at a point \( x \) in the input domain \( \mathcal{D} \) is dictated by the parametric approximation \( \theta_{\mathcal{B}} \) of the subdomain \( \mathcal{B} \) that contains \( x \). We refer to the set of data points that belong to the subdomain \( \mathcal{B} \) as the nearby data for \( x \). The parameters of the subdomain approximation \( \theta_{\mathcal{B}} \) are functions of the data in \( \mathcal{B} \). Therefore the predicted output \( F_c(x) \), equal to \( \theta_{\mathcal{B}}(x) \), is directly assigned by the nearby data for \( x \). By randomly changing the partition \( C \) of the input domain, the size and shape of subdomains containing \( x \) vary and, consequently, so does the set of nearby data for \( x \). Hence the predicted output \( F(x) \) for a point \( x \) in the input domain depends on its nearby data. The deviation of a sample of predicted outputs \( F_c(x) \), represented by \( \sigma_F(x) \), can measure the dependency of the predicted output \( F(x) \) on the nearby data for \( x \).

The deviation measure \( \sigma_F(x) \) computed by the MC-HARP method indicates how sensitive the output predicted by a local approximation \( F_c \) is with respect to the nearby data for \( x \). The MC-HARP method measures the sensitivity of its constructed approximations with respect to the available data, but unlike the sampling performance estimators, it does not hold out any portion of the data. The MC-HARP philosophy is to fit a parametric approximation of fixed structure over a sample of subsets of data points around \( x \) and measure the deviation of outputs predicted among these subsets. The MC-HARP method generates a sample of predicted outputs for a point in the input domain not by randomly holding out some portion of the data, but by assigning different subsets of data close to that point as its nearby data. The MC-HARP method measures the sensitivity of the constructed approximation at a point \( x \) only with respect to data close to \( x \). The performance
measure for \( x \) computed based on sampling techniques represents the sensitivity with respect to the entire data including data points remote from \( x \) and since the behavior of a constructed approximation at a point is not sensitive to data points remote from it, the computed sensitivity is underestimated for each individual point. Including remote data points in the performance estimation and holding some data out of the training process reduce the reliability of the sampling-based techniques in comparison with the MC-HARP method.

The reason that sampling-based techniques include unnecessary remote data points in their deviation measure is that their philosophy is based on the similarities between the problem of mean estimation using a sample of observations and the problem of mapping approximation using a set of data. In their definitions for the approximation deviation, they treat the given data as a sample of separate and discrete observations and they ignore the *spatial neighboring* relations among data points. Therefore their deviation measures do not have any sense of attributes like distance, closeness, and remoteness. On the other hand, the MC-HARP method considers the neighboring relations among data points, and makes the best of it not only for building its approximation but also for estimating a pointwise, approximation deviation measure. One should realize that the sampling-based philosophy for performance estimation can be used for a wide class of statistical problems which may not have any spatial relations among their observations; namely, hypothesis testing and correlation analysis. When there is no spatial relation, the MC-HARP philosophy is not applicable and the sampling-based techniques should be used. However, here we are interested in multivariate mapping approximations and fitting scattered data that have spatial relations among their measurements. Since the MC-HARP method uses the knowledge of spatial relations, it has advantage over the sampling-based techniques that do not consider this important piece of knowledge.

The MC-HARP method uses the spatial relations among data points being of close and neighboring to predict an output that is strongly influenced by the nearby data for that point. The predicted output is the mean of outputs predicted by a sample of locally supported parametric approximations fit over a sample of nearby data sets for that point. Also, the MC-HARP measure of performance sensitivity is the deviation among the sample of locally predicted outputs for that point. The MC-HARP philosophy for building an approximation and for estimating the performance of its constructed approximation is compatible with the generalization goal of building data-based mathematical models. We expect a constructed approximation to generalize for points between data points because the continuity and smoothness of the actual mapping underlying the given data allow the measured outputs for data points to dictate predicted outputs for their neighbor points in the input domain. This dictation, in a mathematical sense, implies interpolation or extrapolation through locally supported, simple functions. The MC-HARP method exactly follows this generalization goal to construct an approximation whose predicted output for a point in the input domain is strongly influenced by its neighboring data. Also the MC-HARP deviation measure \( \sigma_F \) represents how strongly the nearby data for a point influenced its predicted output. The stronger the agreement among a sample of locally supported, simple functions is on the predicted output for a point, the lower the deviation measure \( \sigma_F \) is for that point.

### 5.3.3 A Measure for Confidence in Approximation

We define the approximation deviation measure of \( F \) with optimal complexity to be the *approximation confidence measure*. Considering our definition for the approximation deviation of \( F \), the deviation measure
\[ \sigma_F^{\text{Test}}, \text{ that should be equal to } \sigma_F^{\text{Train}} \text{ for the optimal tolerance value } \varepsilon^*, \text{ is a quantitative measure for the approximation confidence in the constructed approximation } \overline{F}(\varepsilon^*); \text{ i.e., } \\
\] \[ \text{approximation confidence measure } = \frac{1}{\mathcal{D}} \int_\mathcal{D} \sigma_F^{c(\varepsilon^*)}(x) \, dx = \sigma_F^{\text{Test}} \\
\]

We define the confidence measure only for an approximation \( \overline{F} \) whose deviation measures for training and test sets are equal and has subdomains with optimal size and distribution. Furthermore, our definition for the confidence measure implicitly requires the amount of data to be adequate in order to be able to select the optimal tolerance value. The size of subdomains built by the MC-HARP method significantly controls the reliability of \( \sigma_F^{\text{Test}} \) as a confidence measure. For tolerance values greater than \( \varepsilon^* \), subdomains are big and consequently \( \sigma_F \) for a point in \( \mathcal{D} \) represents the approximation sensitivity with respect to remote points and is an underestimate of the value for the approximation deviation. For tolerance values smaller than \( \varepsilon^* \), subdomains are small and consequently \( \sigma_F \) is dominated by noise and is unreliable. The performance measures \( \sigma_F^{\text{Train}} \) is not equal to \( \sigma_F^{\text{Test}} \) for tolerance values relatively smaller than the optimal tolerance value. Therefore the approximation deviation of \( \overline{F} \) is not measurable, and neither is the approximation confidence. Only at the optimal tolerance value the size of subdomains, and consequently the size of nearby data sets for a point in \( \mathcal{D} \), are suitable to compute a reliable, local sensitivity measure with respect to nearby data points and to use the average of pointwise defined sensitivity measures as a reliable measure for the approximation confidence.

We define the deviation among the sample of consistent approximations built by the MC-HARP method for a given data set as a measure of the confidence in approximation. The smaller the deviation measure \( \sigma_F^{\text{Test}} \) is, the higher the approximation confidence is. The confidence in approximation is inversely related to the deviation in approximation. The approximation deviation of \( \overline{F} \) is a measure of the approximation confidence only when the sample of approximations \( \overline{F} \) built by the MC-HARP method are consistent and the amount of data is adequate. The confidence measure is a quantitative measure for the adequacy of data. The higher the approximation confidence is, the better the available set is. From now on, when we can locate the optimal tolerance value on the performance curves \( \sigma_F^{\text{Train}} \) and \( \sigma_F^{\text{Test}} \), we say the amount of data is adequate and we are confident in the constructed approximation. Also, we measure the adequacy of data and the goodness of approximation confidence with the performance measure \( \sigma_F^{\text{Test}} \).

5.4 Accuracy in Approximation

Being able to locate the optimal tolerance on performance curves \( \sigma_F^{\text{Train}} \) and \( \sigma_F^{\text{Test}} \) indicates the data are adequate and one should have great confidence that the constructed approximation has captured all the evident features of the actual mapping underlying the data that were not destroyed by noise. Having good confidence in the approximation does not guarantee that the constructed approximation is acceptably close to the actual mapping because some local features and fine details of the actual mapping require more data to be revealed. Further, the existence of noise in measurements increases the amount of data needed to approximate those fine features. An adequate set of data only contains some features of the actual mapping and, to com-
pletely define the actual mapping, its output value for every point in its input domain should be given. The presence of noise in data significantly increases the amount of data required to approximate the actual mapping for some details.

A measure of distance between the approximated mapping \( \tilde{F} \) and the actual mapping \( G \) is the approximation risk \( RISK_{\tilde{F}} \). The performance measure \( RISK_{\tilde{F}} \) represents the approximation accuracy of \( \tilde{F} \). The smaller is the approximation risk \( RISK_{\tilde{F}} \), the more accurate is the constructed approximation \( \tilde{F} \). The approximation confidence measure \( \sigma_{F_{\epsilon}(\epsilon^*)} \) and the approximation accuracy measure \( RISK_{F_{\epsilon}(\epsilon^*)} \) are the mean average of standard deviations and biases of samples of predicted outputs \( \{F_{c_{\epsilon}(\epsilon^*)}(x)\}_{i=1}^{p} \) for all \( x \) in the input domain, respectively. The deviation measure \( \sigma_{F_{\epsilon}(\epsilon^*)} \) is the mean distance between an approximation in the approximation sample \( \{F_{c_{\epsilon}(\epsilon^*)}\}_{i=1}^{p} \) and the sample mean average \( \bar{F}(\epsilon^*) \). The bias measure \( RISK_{F_{\epsilon}(\epsilon^*)} \) is the distance between the sample mean average \( \bar{F}(\epsilon^*) \) and the actual mapping \( G \).

The confidence measure represents a performance quantity different from the accuracy measure. However we know that the deviation measure \( \sigma_{F} \), computed by the MC-HARP method, follows the complexity of data. For regions in the input domain where the complexity of the actual mapping is higher than the assumed subdomain approximation \( \theta \) and the approximation accuracy is lower than other regions in \( \mathcal{D} \), the deviation measure \( \sigma_{F} \) is larger than for other regions with higher accuracy and lower data complexity. Also by increasing the amount of data, both the confidence measure \( \sigma_{F_{\epsilon}(\epsilon^*)} \) and the accuracy measure \( RISK_{F_{\epsilon}(\epsilon^*)} \) converge to zero. Finally it is evident in Figs. 4.2 through 4.5 that the deviation measure \( E\sigma_{F_{\epsilon}(\epsilon^*)} \) is close to the approximation risk \( RISK_{\tilde{F}} \) for tolerance values close to the optimal tolerance \( \epsilon^* \) when the amount of data is adequate. (In our numerical simulations, for tolerance values distant from \( \epsilon^* \) or inadequate data sets, the deviation measure \( E\sigma_{F_{\epsilon}(\epsilon^*)} \) generally underestimates the approximation risk \( RISK_{\tilde{F}} \). This is another reason for using adequate data and constructed approximations with optimal complexity. Figure 5.4 shows the variation of the approximation risk \( RISK_{F_{\epsilon}(\epsilon^*)} \) and the approximation deviation measure \( E\sigma_{F_{\epsilon}(\epsilon^*)} \) with respect to the number of data for different amplitudes of noise. It is evident that by increasing the number of data, the deviation measure gets closer to the approximation risk and for large amounts of data, it is greater than the accuracy measure \( RISK_{F_{\epsilon}(\epsilon^*)} \). For a fixed amplitude of noise there is a critical amount of data that, for data sets containing more data than this critical amount, the deviation measure is greater than or equal to the approximation risk. This critical amount of data increases as the noise amplitude increases. For adequate data sets, the confidence measure \( E\sigma_{F_{\epsilon}(\epsilon^*)} \) is equal to or greater than the accuracy measure \( RISK_{F_{\epsilon}(\epsilon^*)} \).

In the function space, the batch (sample) of optimal approximations \( \{F_{c_{\epsilon}(\epsilon^*)}\}_{i=1}^{p} \) built by the MC-HARP method approaches the actual mapping \( G \) such that its mean radius \( E\sigma_{F_{\epsilon}(\epsilon^*)} \) decreases as the number of data increases. Also the mean radius of the batch of solutions is greater than the distance between the batch center \( \bar{F}(\epsilon^*) \) and the actual mapping \( G \) when the amount of data is adequate. In other words, for adequate data sets, the batch of solutions covers \( G \) and the deviation measure of the sample of solutions is greater than its bias value. These characteristics recommend that the confidence measure \( E\sigma_{F_{\epsilon}(\epsilon^*)} \) is a useful upper bound for the approximation accuracy measure \( RISK_{F_{\epsilon}(\epsilon^*)} \). To have a reliable measure for the approximation confidence and accuracy, the amount of data should be adequate to locate the optimal tolerance value \( \epsilon^* \) on the performance curves \( \sigma_{F_{\epsilon}}^{train} \) and \( \sigma_{F_{\epsilon}}^{test} \). The adequacy of data does not infer that the constructed approximation has
acceptable accuracy, but it does indicate that the computable confidence measure is a reasonable approximation for the approximation accuracy. If the amount of data is adequate and the confidence measure is acceptable, then the constructed approximation by the MC-HARP method has an acceptable accuracy.

5.5 Confidence Intervals

Since the deviation measure $\sigma_{F(e^*)}^{\text{Test}}$ is a reasonable approximation for the accuracy of the constructed mapping $\bar{F}(e^*)$, the deviation measure $\sigma_{F(e^*)}(x)$ for a point $x$ in the input domain $\mathcal{D}$ can be a reasonable approximate for the accuracy of the predicted output of $\bar{F}(e^*)$ for $x$. An approximate confidence interval for an unknown actual output $G(x)$ can be given as follows

$$|G(x) - \bar{F}(x)| \leq z(\alpha)\sigma_{F(e^*)}(x) \quad \forall x \in \mathcal{D} \tag{5.5}$$

where $\bar{F}(x)$ is the predicted output of $\bar{F}(e^*)$ for a point $x$ and $z(\alpha)$ is the length of the $100\alpha$ percent central confidence interval for a standard normal variate; e.g., $z(0.95) = 1.96$. Our numerical simulations have shown that for adequate data sets, the deviation measure $E_{\sigma_{F(e^*)}}^{\text{Test}}$, which is the mean average of pointwise computed deviation values $\sigma_{F(e^*)}(x)$ for all $x$ in $\mathcal{D}$, is greater than the bias measure $E_{\text{ERISK}} F(e^*)$, which is the square root of the mean average of pointwise computed squared bias values $(G(x) - \bar{F}(x))^2$. Therefore the deviation value $\sigma_{F(e^*)}(x)$ can be a reasonable upper bound for the absolute error $|G(x) - \bar{F}(x)|$ and the confidence interval defined in Eqn. (5.5) with $z(\alpha)$ greater than one is conservative on average for all points in the input domain.

A more sophisticated way to approximate confidence intervals is to use the empirical distribution of the random variable $F_{C(e^*)}(x)$ instead of the normal distribution pre-assumed in Eqn. (5.5). Because of the randomness of partitions $C$ generated by the MC-HARP method, the predicted output $F_{C(e^*)}(x)$ is a random variable. Define $g^*(u; x)$ to be the empirical cumulative distribution function of $F_{C(e^*)}(x)$. 

---

Fig. 5.4 Variations of the approximation risk and the approximation deviation for the optimal tolerance value
(a) Actual function $G_1$, (b) Actual function $G_2$
\[ g^*(u; x) = \frac{\#\{F_{c(e^*)}(x) \leq u\}_{i=1}^p}{p} \]  

where \( p \) is the number of random partitions generated by the MC-HARP method and the operator \# counts the number of members in a set. The value \( g^*(u; x) \) is an approximation for the probability of a predicted output \( F_{c(e^*)}(x) \) being less than or equal to \( u \). The 100\( \alpha \) percent confidence interval for an unknown actual value \( G(x) \) is defined as

\[ g^*-1\left(\frac{1-\alpha}{2}; x\right) \leq G(x) \leq g^*-1\left(\frac{1+\alpha}{2}; x\right) \quad \forall x \in \mathbb{D} \]  

where \( g^*-1(\beta; x) \) is the 100\( \beta \) percentile point of the random variable \( F_{c(e^*)}(x) \); i.e., \( g^*(g^*-1(\beta; x); x) = \beta \). To compute the confidence intervals in Eqn. (5.5), only the deviation value \( \sigma_{F(e^*)}(x) \) should be known, but for the confidence intervals in Eqn. (5.7), the sample of predicted outputs \( \{F_{c(e^*)}(x)\}_{i=1}^p \) are needed to calculate the percentile points for the empirical distribution of predicted outputs. Hence the confidence intervals defined in Eqn. (5.7) are computationally more expensive but statistically more customized than the standard intervals defined in Eqn. (5.5). The defined confidence intervals are useful tools when the amount of data is adequate, otherwise they can be quite inaccurate.

We have already defined a model selection criterion, a quantitative measure for the approximation confidence, an objective measure for the data adequacy, and confidence bounds for the approximation accuracy. All of these definitions are based on the performance curves \( \sigma_{F}^{Train} \) and \( \sigma_{F}^{Test} \). There is more information in these performance curves than what we have already extracted. In following sections, we establish a framework for classifying data-fitting problems and evaluate it using a set of real data.

### 5.6 Critical Points on the MC-HARP Computable Performance Curves

For a given data set, one can always compute the performance measure \( RMS_{F}^{Train} \) for a constructed approximation \( F \). The MC-HARP method can provide two additional performance measures, \( \sigma_{F}^{Train} \) and \( \sigma_{F}^{Test} \), for its constructed mappings. By changing the tolerance value \( \varepsilon \), one can always compute the three performance measures for the family of approximations \( \overline{F} \) that are built by the MC-HARP method and have different complexity. In other words, the performance curves \( RMS_{F}^{Train} \), \( \sigma_{F}^{Train} \) and \( \sigma_{F}^{Test} \) can be computed for any given data set. From now on, we refer to these curves as the \textit{MC-HARP computable performance curves}. The performance measures that need values of the actual mapping like the approximation risk \( ERISK_{F} \) are not computable for actual data sets. Furthermore, one can compute statistical performance measures like cross-validation, bootstrap, or predicted squared error for a given data set.

Our main goal is to establish a framework of rules to assist an experimenter in selecting the best possible approximation built by the MC-HARP method and to classify his data-fitting problem based on objective measures for the quantity and quality of the given data set. To accomplish our agenda, we will first define a few specific points that we refer to as \textit{critical points} on the MC-HARP computable performance curves. Then we will use these defined critical points to set a framework of rules for selecting the best tolerance value \( \varepsilon \) that
should be as close as possible to the optimal tolerance value $\varepsilon^*$. Meanwhile, we will establish a *quantity-quality map* to classify data-based mapping approximation problems.

The critical points on the MC-HARP computable performance curves are distinguished points on the performance curves $\text{RMS}_{\text{Train}}^\text{Test}$, $\sigma_{\text{Train}}^\text{Train}$ and $\sigma_{\text{Test}}^\text{Test}$ that define the behavior of these performance curves for different amounts of data and amplitudes of noise. We call these critical points by their corresponding tolerance value; *i.e.*, critical points $\varepsilon_t$ through $\varepsilon_s$. Our observations based on numerical simulations and the model selection philosophy of the MC-HARP method motivate us to select these critical points among all points on the computable performance curves. In this section, we define these critical points, explain their importance, and give hints to locate them.

### 5.6.1 Critical Point $\varepsilon_o$

We start with the critical tolerance value $\varepsilon_o$ that represents the upper bound of the interval $[0, \varepsilon_o]$ at which the deviation measure $\sigma_{\text{Test}}^\text{Test}$ is steady and constant. The definition of $\varepsilon_o$ can be mathematically written as:

$$
\varepsilon_o = \max \left\{ \xi : 0 \leq \xi \leq \varepsilon_{\text{max}} , \forall a, b \in [0, \xi], \sigma_{\text{Test}}^\text{Test}(N, a, \lambda) = \sigma_{\text{Test}}^\text{Test}(N, b, \lambda) \right\}
$$

where the approximation sign in Eqn. (5.8) means that the deviation measure $\sigma_{\text{Test}}^\text{Test}$ does not change significantly. At $\varepsilon = \varepsilon_o$, the performance curve $\sigma_{\text{Test}}^\text{Test}$ finishes its initial constant behavior and starts to increase or decrease, as shown in Figs. 4.2 through 4.5. By decreasing the amount of noise, the tolerance value $\varepsilon_o$ converges to zero and for a small amount of noise, the performance curve $\sigma_{\text{Test}}^\text{Test}$ is non-constant for small tolerance values. If $\varepsilon_o$ exists, by increasing the amount of data, it reaches its limit value $\hat{\varepsilon}_o$ and does not change significantly after that.

By increasing the tolerance value from zero to $\varepsilon_o$, the deviation measure $\sigma_{\text{Test}}^\text{Test}$ does not change, but $\sigma_{\text{Train}}^\text{Train}$ increases indicating that as $\varepsilon$ increases, the deviation measure $\sigma_{\text{F}}$ increases around training data points but does not change for points far from the training data points. When the amplitude of noise is large and the tolerance value is small, the subdomains constructed by the HARP method are small and their corresponding, local, parametric, subdomain approximations are dominated by noise. Consequently, the approximation deviation $\sigma_{\text{F}}$, for points in between training data points, are controlled by noise and do not change until the sizes of subdomains become large enough to contain the amount of data representing a local feature separable from added noise. Therefore, for the range of tolerance values $[0, \varepsilon_o]$, the deviation measure $\sigma_{\text{Test}}^\text{Test}$ does not change when the amount of noise is large.

Because of the noise dominance effect, the approximation risk decreases or does not change significantly by varying the tolerance value from zero to $\varepsilon_o$. Hence $\varepsilon_o$ is a reasonable lower bound for the selected best tolerance value $\varepsilon_s$. In other words, the constructed approximation $\widetilde{F}(\varepsilon_o)$ is the most complex approximation that one should select from the family $\mathcal{F}$. By selecting approximations $\widetilde{F}(\varepsilon)$ whose tolerance values are smaller than $\varepsilon_o$, we increase the number of parameters in the constructed approximation; *i.e.*, complexity, without decreasing the approximation risk and often with the danger of increasing the risk from its value at $\varepsilon_o$. 

120
5.6.2 Critical Point $\varepsilon_1$

The critical tolerance value $\varepsilon_1$ locates the intersection of the $\text{RMS}^\text{Train}_F$ and $\sigma^\text{Train}_F$ performance curves. For the tolerance value $\varepsilon_1$ greater than zero, the following equality can be written

$$
\sigma^\text{Test}_F(N, \varepsilon_1, \lambda) = \text{RMS}^\text{Train}_F(N, \varepsilon_1, \lambda)
$$

The critical point $\varepsilon_1$ represents the complexity at which the deviation of constructed approximations by the MC-HARP method is equal to the approximation error for the training set. Generally, for tolerance values greater than $\varepsilon_1$, the deviation measure $\sigma^\text{Train}_F$ is smaller than the approximation error $\text{RMS}^\text{Train}_F$. We expect that, for a subinterval of $[0, \varepsilon_{\text{max}}]$, the performance measure $\text{RMS}^\text{Train}_F$ is greater than the performance measure $\sigma^\text{Train}_F$ because, as we have mentioned in Section 4.5, the performance curve $\text{RMS}^\text{Train}_F$ is a non-decreasing curve and the performance curve $\sigma^\text{Train}_F$ has zero end points. Therefore, there is at most one intersection between $\sigma^\text{Train}_F$ and $\text{RMS}^\text{Train}_F$ performance curves besides their common starting point. For large amplitudes of noise, the critical point $\varepsilon_1$ generally exists. For small amplitudes of noise, it may not exist. When $\varepsilon_1$ exists, it converges to a limit value $\hat{\varepsilon}_1$ as the amount of data increases. The critical tolerance value $\varepsilon_1$ converges to its limit value faster for large amounts of noise than for small amounts of noise.

The optimal tolerance value $\varepsilon^*$ is greater than the limit value $\hat{\varepsilon}_1$, as shown in Figs. 4.2 through 4.5. For large amplitudes of noise, the critical tolerance value $\varepsilon_1$ reaches its limit value even for moderate data sets and $\varepsilon^*$ is greater than $\hat{\varepsilon}_1$. For small amplitudes of noise, $\varepsilon^*$ may be smaller than $\varepsilon_1$, if it exists, but by increasing the amount of data, $\varepsilon_1$ converges to its limit value $\hat{\varepsilon}_1$ that is smaller than $\varepsilon^*$. The reason that $\varepsilon^*$ is greater than $\hat{\varepsilon}_1$ can be explained as follows: a local minimum is developed on the performance curve $\sigma^\text{Train}_F$ at $\varepsilon^*$ as the number of data increases. The minimum value of $\sigma^\text{Train}_F$ computed at $\varepsilon^*$ converges to zero by increasing the amount of data. On the other hand, the approximation error $\text{RMS}^\text{Train}_F$ is a combination of two measures: the standard deviation of noise and the bias between the constructed approximation $F$ and the actual mapping. By increasing the amount of data, for approximations with tolerance values close to $\varepsilon^*$ the bias measure converges to small values but the standard deviation of noise cannot be reduced. Therefore the performance curve $\text{RMS}^\text{Train}_F$ converges to values greater than zero and close to the standard deviation of noise for the tolerance values close to $\varepsilon^*$. Consequently, the performance measure $\text{RMS}^\text{Train}_F$ becomes greater than the deviation measure $\sigma^\text{Train}_F$ for the tolerance values close to $\varepsilon^*$ as the amount of data increases. Because for the tolerance values greater than $\varepsilon_1$ the measure $\text{RMS}^\text{Train}_F$ is greater than $\sigma^\text{Train}_F$, the optimal tolerance value $\varepsilon^*$ is greater than the limit value $\hat{\varepsilon}_1$.

For noiseless data, the optimal tolerance value $\varepsilon^*$ is equal to zero and consequently no local minimum is developed on the performance curve $\sigma^\text{Train}_F$ within its support $[0, \varepsilon_{\text{max}}]$. The shapes of performance curves $\sigma^\text{Train}_F$ and $\text{RMS}^\text{Train}_F$ reach their limits for moderate sets of data and do not change afterwards as the amount of data increases. By increasing the amount of data, among three computable performance curves, only the initial portion of the performance curve $\sigma^\text{Test}_F$ changes and converges to zero. In this case, if there exists a critical tolerance $\varepsilon_1$ that represents the intersection of performance curves $\sigma^\text{Train}_F$ and $\text{RMS}^\text{Train}_F$, it does not con-
verge to zero and remains greater than zero for large data sets. Hence for noiseless data the optimal tolerance value \( \varepsilon^* \), which is equal to zero, remains less than \( \varepsilon_1 \), if it exists, and its limit value \( \hat{\varepsilon}_1 \).

Another reason for \( \varepsilon^* \) being greater than \( \hat{\varepsilon}_1 \) can be explained as follows: the deviation measure \( \sigma_F \) represents the precision of the approximation built by the MC-HARP method. For a consistent approximation like \( \bar{F}(\varepsilon^*) \), its precision should be greater than the precision of the available data in order to capture the features contained in the data and not to be dominated by noise. One can draw an analogy between the problem of building data-based mathematical models and the problem of designing a measuring device. In a data-fitting problem, we want to build a tool that represents the features contained in data disturbed by noise. In a measuring device problem, we want to design a tool to measure some attributes that are contaminated with noise. A measuring device whose precision is less than the size of the quantities that we want to measure is not precise enough to make the measurement. For example, a thermometer whose precision is one degree should not be used to measure a temperature value with noise fluctuations whose amplitudes are less than one degree because it is possible to measure this temperature value with better precision than our thermometer is capable.

The same reasoning is true for building approximations with the MC-HARP method. The precision of constructed approximations is measured by the performance measure \( \sigma_F \). For the optimal tolerance value \( \hat{\varepsilon}_1 \) the MC-HARP approximations are consistent, meaning that they can converge to the actual mapping with any accuracy if there are enough data. In other words, the MC-HARP approximations can represent the actual mapping with high accuracy. Since our constructed tool, the MC-HARP approximation \( \bar{F}(\varepsilon^*) \), can capture the features of the actual mapping contained in the data with any required accuracy, its precision should be greater than the precision of the available data. Therefore, the deviation measure of the MC-HARP approximations \( \sigma_{FC\varepsilon_1} \) becomes smaller than the standard deviation of noise as the amount of data increases, otherwise the MC-HARP approximations \( FC(\varepsilon_1) \) are not precise enough to approximate the actual mapping with any required accuracy and thus they are not consistent and reliable tool for approximation.

The consistency characteristic requires the precision of constructed mappings to be greater than the deviation of noise in the data. So for the training set, by increasing the amount of data, the deviation measure \( \sigma_F^{\text{Train}} \) becomes smaller than the standard deviation of noise for tolerance values close to the optimal tolerance value \( \varepsilon^* \). Since the approximation error \( RMS_F^{\text{Train}} \) is a combination of the bias in approximation and the standard deviation of noise, therefore for tolerance values close to \( \varepsilon^* \), the deviation measure \( \sigma_F^{\text{Train}} \) definitely becomes smaller than the measure \( RMS_F^{\text{Train}} \) as the amount of data increases. Hence \( \varepsilon^* \) is greater than \( \hat{\varepsilon}_1 \) for sufficiently large, noisy data sets.

### 5.6.3 Critical Point \( \varepsilon_2 \) and \( \varepsilon_4 \)

As we mentioned in Section 4.5, since the performance curve \( \sigma_F^{\text{Train}} \) represents a nonconstant, positive function with respect to \( \varepsilon \), and is equal to zero at the end points of its domain interval, it should have at least one maximum point at some tolerance value, as shown in Figs. 4.2 through 4.5. The critical tolerance value \( \varepsilon_2 \) represents the location of the primary maximum of the performance curve \( \sigma_F^{\text{Train}} \). The definition of \( \varepsilon_2 \) can be written as:
\[ \exists \beta > 0 : \forall \varepsilon \in (\varepsilon_2 - \beta, \varepsilon_2 + \beta) \quad \sigma^\text{Train}_F (N, \varepsilon_2, \lambda) \geq \sigma^\text{Train}_F (N, \varepsilon, \lambda) \quad (5.10) \]

The reason that we refer to the maximum of \( \sigma^\text{Train}_F \) curve corresponding to \( \varepsilon_2 \) as the primary one is that a secondary maximum can be developed on this curve by increasing the amount of data. For the optimal tolerance value \( \varepsilon^* \), a local minimum is developed on the performance curve \( \sigma^\text{Train}_F \) as the number of data increases. The development of the local minimum dictates the development of two maximum points on both sides of \( \varepsilon^* \) in the domain interval \([0, \epsilon_{\text{max}}]\) of the \( \sigma^\text{Train}_F \) curve. We refer to the maximum point that exists before the development of the local minimum as the primary maximum. The other maximum point is the secondary maximum and we refer to its location as the critical point \( \varepsilon_4 \).

The critical tolerance \( \varepsilon_2 \) exists for any data set that cannot be fit perfectly; i.e., with tolerance value equal to zero, by the subdomain approximation \( \theta \). So the primary maximum exists for moderate data sets, but the secondary maximum cannot be located until the amount of data is large enough to develop a local minimum on the performance curve \( \sigma^\text{Train}_F \). Both critical tolerance values \( \varepsilon_2 \) and \( \varepsilon_4 \) have limit values \( \hat{\varepsilon}_2 \) and \( \hat{\varepsilon}_4 \) respectively as the amount of data increases. Since the local minimum at \( \varepsilon^* \) is developed on the slopes of the primary peak of the performance curve \( \sigma^\text{Train}_F \), the secondary maximum value \( \sigma^\text{Train}_F (N, \varepsilon_4, \lambda) \) is generally smaller than the primary maximum value \( \sigma^\text{Train}_F (N, \varepsilon_2, \lambda) \). In other words, the primary maximum is the global maximum and in Eqn. (5.10) the interval \((\varepsilon_2 - \beta, \varepsilon_2 + \beta)\) can be replaced by the interval \([0, \epsilon_{\text{max}}]\).

**Remarks about two important special cases.** For the noiseless data sets, since the optimal tolerance \( \varepsilon^* \) is equal to zero, its corresponding local minimum on \( \sigma^\text{Train}_F \) curve is developed at \( \varepsilon = 0 \). In this case, the primary maximum is on the right side of \( \varepsilon^* \) and no secondary maximum point can exist on the left side of \( \varepsilon^* \). There is only one maximum point for any given data set. The same situation happens when the actual mapping \( G \) belongs to the family \( \Theta \) of parametric, subdomain approximations \( \theta \). In this case, the optimal tolerance value \( \varepsilon^* \) is equal to \( \epsilon_{\text{max}} \) and no secondary maximum can be developed on the right side of \( \varepsilon^* \). Hence in the case that the given data is noiseless or the actual function \( G \) belongs to the family \( \Theta \), no secondary maximum develops on the performance curve \( \sigma^\text{Train}_F \) as the amount of data increases. This characteristic can be used to check that the given data is noise-free or not and to verify that a presumed parametric approximation has the correct structure to represent the actual mapping.

One can test the hypothesis that the given data is noiseless by increasing the amount of training data presented to the MC-HARP method. If a secondary peak is developed on the performance curve \( \sigma^\text{Train}_F \) then the given data set is noisy, otherwise, it is noiseless. Furthermore, to verify that the actual mapping can be accurately represented by a presumed parametric function, one can select the subdomain approximation \( \theta \) to be the presumed parametric function. Then, if by increasing the amount of data no secondary maximum is developed on the \( \sigma^\text{Train}_F \) curve, the assumed parametric form can represent the actual mapping exactly, otherwise, the actual mapping is more complex than the assumed parametric form.

**5.6.4 Critical Point \( \varepsilon_3 \)**

The performance curve \( \sigma^\text{Test}_F \) approaches the \( \sigma^\text{Train}_F \) curve and, for tolerance values greater than the critical tolerance value \( \varepsilon_3 \), \( \sigma^\text{Test}_F \) is either equal or close to \( \sigma^\text{Train}_F \). The definition of \( \varepsilon_3 \) can be written as...
\[ \varepsilon_3 = \min \{ \varepsilon : \forall \beta \in [\varepsilon, \varepsilon_{\text{max}}] \quad \sigma_F^{\text{Train}}(N, \beta, \lambda) = \sigma_F^{\text{Test}}(N, \beta, \lambda) \} \]  

(5.11)

The critical point \( \varepsilon_3 \) can be located in Figs. 4.2 through 4.5. When the data set is not large enough, the performance curve \( \sigma_F^{\text{Test}} \) may be separate from the \( \sigma_F^{\text{Train}} \) curve, except for its end region that often has a steep drop. In this case, we say that the critical point \( \varepsilon_3 \) does not exist. When \( \varepsilon_3 \) exists, the performance curve \( \sigma_F^{\text{Test}} \) is close to the \( \sigma_F^{\text{Train}} \) curve for a significant portion of its support. By increasing the amount of data, the critical tolerance value \( \varepsilon_3 \) converges to a limit value \( \hat{\varepsilon}_3 \). For the optimal tolerance \( \varepsilon^* \), the performance measures \( \sigma_F^{\text{Train}} \) and \( \sigma_F^{\text{Test}} \) become equal to each other and converge to zero as the number of data increases. Hence, based on the definition of \( \varepsilon_3 \), the optimal tolerance \( \varepsilon^* \) belongs to the interval \( [\hat{\varepsilon}_3, \varepsilon_{\text{max}}] \) and is generally close to \( \hat{\varepsilon}_3 \).

For tolerance values close to \( \varepsilon_{\text{max}} \), the subdomains in a random partition \( \mathcal{C} \) are large and the probability that a training data point and its nearby test data points belong to the same subdomain is high. Hence the predicted output for a training data point and its nearby test points are computed by the same subdomain approximation \( \theta_C \) with high probability. Consequently, the deviation of predicted outputs for a training data point, computed by a sample of subdomain approximations \( \theta_C \) corresponding to a sample of random partitions \( \mathcal{C} \) of the input domain, is close to the deviation of predicted outputs for its nearby test data points that are computed with high probability using the same sample of subdomain approximations \( \theta_C \). Therefore for approximations \( F_C(\varepsilon) \) with low complexity, whose \( \varepsilon \) is close to \( \varepsilon_{\text{max}} \), the deviation measure \( \sigma_F^{\text{Test}} \) for the test set can become equal to the deviation measure \( \sigma_F^{\text{Train}} \) for the training set. On the other hand, for tolerance values close to zero, the subdomains are small and the probability that a training data point and its nearby test data points belong to the same subdomain is low. Consequently, the deviation measure \( \sigma_F^{\text{Test}} \) of predicted outputs for the test data points cannot become equal to the deviation measure \( \sigma_F^{\text{Train}} \) for constructed approximations \( F_C(\varepsilon) \) with high complexity, whose \( \varepsilon \) is close to zero.

The performance curves \( \sigma_F^{\text{Train}} \) and \( \sigma_F^{\text{Test}} \) are completely separate from each other at \( \varepsilon = 0 \) where the \( \sigma_F^{\text{Train}} \) measure is equal to zero and the \( \sigma_F^{\text{Test}} \) measure is greater than zero. These curves can join for tolerance values close to \( \varepsilon_{\text{max}} \). Hence, when the amount of data is not small, the critical point \( \varepsilon_3 \), which represents the meeting point of the performance curves \( \sigma_F^{\text{Train}} \) and \( \sigma_F^{\text{Test}} \), exists. For noiseless data, the critical tolerance value \( \varepsilon_3 \) converges to zero; i.e., \( \hat{\varepsilon}_3 = 0 \), by increasing the amount of data because the deviation measure \( \sigma_F^{\text{Test}} \) at \( \varepsilon = 0 \) converges to zero and becomes equal to the \( \sigma_F^{\text{Train}} \) measure. In this case, the optimal tolerance value \( \varepsilon^* \) that is equal to zero is out of the interval \( [\varepsilon_3, \varepsilon_{\text{max}}] \) but is in the interval \( [\hat{\varepsilon}_3, \varepsilon_{\text{max}}] \) since \( \hat{\varepsilon}_3 = \varepsilon^* = 0 \). For noiseless data, the training set should be infinitely large to allow \( \varepsilon_3 \) to reach its limit value \( \hat{\varepsilon}_3 \) and one should know that the optimal tolerance \( \varepsilon^* \) is smaller than \( \varepsilon_3 \) for any given data set. For noisy data, the critical tolerance \( \varepsilon_3 \) reaches its limit value \( \hat{\varepsilon}_3 \) for an adequate amount of data. The optimal tolerance value \( \varepsilon^* \) is greater than \( \hat{\varepsilon}_3 \) and, with good probability, than \( \varepsilon_3 \) computed for the given noisy data set.

5.6.5 Critical Point \( \varepsilon_5 \)

When the amount of data is adequate, a local minimum is developed on both performance curves \( \sigma_F^{\text{Train}} \) and \( \sigma_F^{\text{Test}} \) for the optimal tolerance \( \varepsilon^* \) and we refer to such a local minimum as a mature local minimum. For
moderate but inadequate data sets, a local minimum may be developed on only one of these performance curves and we refer to this local minimum as a *premature local minimum*. The tolerance value corresponding to a premature local minimum is a good candidate for the optimal tolerance value \( \varepsilon^\ast \). The critical point \( \varepsilon_5 \) represents the location of such a premature local minimum on the performance curve \( \sigma_F^{Test} \). The definition of \( \varepsilon_5 \) can be written as

\[
\exists \beta > 0 : \forall \varepsilon \in (\varepsilon_5 - \beta, \varepsilon_5 + \beta) \quad \sigma_F^{Test}(N, \varepsilon_5, \lambda) \leq \sigma_F^{Test}(N, \varepsilon, \lambda)
\]

We have observed the development of the critical point \( \varepsilon_5 \) on the \( \sigma_F^{Test} \) curve during our numerical simulation, as shown in Fig. 4.4 for \( \lambda=0.05 \).

When the amplitude of noise is small, the optimal tolerance \( \varepsilon^\ast \) is close to zero. For tolerance values close to zero and \( \varepsilon^\ast \), the deviation measure \( \sigma_F^{Train} \) is close to zero, but the deviation measure \( \sigma_F^{Test} \) is greater than zero. The amount of data needed to decrease \( \sigma_F^{Test} \) to the value of \( \sigma_F^{Train} \) for the optimal tolerance may be significantly large. Hence there is a possibility that a local minimum is developed on the \( \sigma_F^{Test} \) curve before it meets the \( \sigma_F^{Train} \) curve. The critical point \( \varepsilon_5 \) that locates such a premature local minimum may exist for a given data set. By increasing the amount of data, the critical tolerance \( \varepsilon_5 \) converges to the optimal tolerance \( \varepsilon^\ast \); i.e., \( \hat{\varepsilon}_5 = \varepsilon^\ast \), and the premature local minimum becomes a mature local minimum.

### 5.7 Quality and Quantity Estimation of Data

The characteristics of noise added to the given data are defined by its distribution. For real-world problems, the distribution of noise is generally unknown but noise exists. An objective measure for the quality of data is the amplitude of noise defined for each data point or the entire data set. An estimation of the noise amplitude can help an experimenter to examine the quality of the given data and the reliability of the constructed approximation. There are statistical estimates of the amplitude of noise using statistics computed for residuals between outputs predicted by the constructed mapping and the expected outputs. The estimates like the cross validation, bootstrap, and predicted squared error for the prediction error can be used to estimate the noise amplitude. The statistical estimates are reliable when the constructed mapping is close to the actual mapping and the assumed distribution for noise is close to its actual distribution. For MC-HARP, one can use statistical estimates for the constructed MC-HARP approximation with optimal complexity, \( F(\varepsilon^\ast) \), to compute a reliable estimate for the amplitude of noise. Since \( F(\varepsilon^\ast) \) has captured main features of the actual mapping contained in the given data, the estimates of the noise amplitude using the residuals between outputs predicted by \( F(\varepsilon^\ast) \) and the actual mapping are statistically reliable. To select the optimal tolerance value \( \varepsilon^\ast \), the amount of training data needs to be adequate; i.e., \( N \geq N^\ast \). In this section we introduce how one can use the performance curves of MC-HARP to qualitatively estimate the amplitude of noise even when the amount of data is not adequate but is larger than a required minimum \( N_{\min} \).

First we need to define the *minimal amount of training data* \( N_{\min} \). For the data sets containing more than \( N_{\min} \), the values and locations of maximum points of the performance curves \( RMS_F^{Train} \) and \( \sigma_F^{Train} \) do not vary significantly by increasing the number of data \( N \). We define \( N_{\min} \) such that for \( N \geq N_{\min} \),
$\text{RMS}_{\text{Train}}^T(N, \varepsilon_{\text{max}} \lambda), \varepsilon_{\text{max}}(N, \lambda), \sigma_{\text{Train}}^T(N, \varepsilon_{2} \lambda)$, and $\varepsilon_{2}(N, \lambda)$ do not vary significantly by increasing the amount of data and are close to their limit values. Since, for $\varepsilon = \varepsilon_{\text{max}}$, the MC-HARP approximation is equal to the subdomain approximation $\theta$, the invariance of $\varepsilon_{\text{max}}$ and $\text{RMS}_{\text{Train}}^T(N, \varepsilon_{\text{max}} \lambda)$ for data sets with more data than $N_{\text{min}}$ means that the parameters of a parametric function $\theta$ converge to their limit values. For example, when a constant subdomain approximation $\theta$ reaches its limit value, it means that there are enough data in the given data set to compute the mean average of the actual mapping for the entire input domain assuming that the least-squares estimator is used to estimate the parameter of the constant function $\theta$. In this case, the $\varepsilon_{\text{max}}$ and $\text{RMS}_{\text{Train}}^T(N, \varepsilon_{\text{max}} \lambda)$ represent the maximum and the mean squared distance between the noisy actual mapping and the mean average of the actual mapping within the entire input domain. A given data set should at least contain enough data to represent the mean average of the actual mapping. A data set that does not contain this minimal amount of data, almost surely, contains no valuable information about the actual mapping.

The reasoning for defining $N_{\text{min}}$ is that a data set for a partitioning-based method like MC-HARP should at least contain enough data to reliably fit a parametric approximation like $\theta$. When the amount of data is greater than this required minimal amount then we can partition the input domain and use more local parametric approximations to improve the accuracy of fit.

For data sets with $N \geq N_{\text{min}}$, the end points and peak points of the performance curves $\text{RMS}_{\text{Train}}^T$ and $\sigma_{\text{Train}}^T$ reach their limit values. In other words, the general shapes of these performance curves are established for data sets containing more data than $N_{\text{min}}$ and their shapes locally change only around the optimal tolerance value $\varepsilon^*$ by providing more data. Using the definition of the minimal amount of data $N_{\text{min}}$, we have schematically shown the curve $N_{\text{min}}(\lambda)$ in Figs. 5.1 and 5.2. Since a noise-free data set should contain enough data to at least represent the global features of the actual mapping including its mean average over the entire input domain, $N_{\text{min}}(0)$ is greater than zero. From sampling theory we know that the standard error between a sample mean average and the actual expected value is an increasing function of the variance of the random variable and decays with respect to the sample size. Therefore, by increasing the variance, the sample size should be increased in order to estimate the expected value with the same confidence. Similarly for the data based mapping approximation, the amount of data (sample size) should be increased with the variance of noisy, expected outputs in order to compute the mean average of the actual mapping with good confidence. Hence the minimal amount of data $N_{\text{min}}$ increases with the variance of expected outputs. The function $N_{\text{min}}(\lambda)$ is an increasing function of the noise variance $\lambda^2$, as shown in Figs. 5.1 and 5.2.

One should notice that the $N_{\text{min}}(\lambda)$ curve is not as crisp as shown. It only represents the fuzzy boundary between the noisy data sets containing minimal amount of data and those that do not. The minimal amount of data should be adequately greater than the number of parameters in the subdomain approximation $\theta$ in order that the estimated parameters of $\theta$ have good confidence and consequently, the global performance measures $\varepsilon_{\text{max}}, \text{RMS}_{\text{Train}}^T(N, \varepsilon_{\text{max}} \lambda), \sigma_{\text{Train}}^T(N, \varepsilon_{2} \lambda)$, and $\varepsilon_{2}$ reach their limit values. Furthermore, as we have shown, the curse of dimensionality increases the minimal amount of data required to represents the global features including mean average, maximum error, and maximum deviation of the actual mapping. Thus, the sparseness caused by the dimensionality of data increases the value of $N_{\text{min}}$. 

126
Another measure for the quantity of data besides $N_{\text{min}}$ is the \textit{adequate amount of data} $N^*$ required to represent the main, local features of the actual mapping that have not been destroyed by noise. We say the amount of data is adequate; \textit{i.e.}, $N \geq N^*$, when we can locate the optimal tolerance value on the performance curves $\sigma_F^{\text{Test}}$ and $\sigma_F^{\text{Train}}$. In Figs. 5.1 and 5.2, we have schematically shown the fuzzy boundary between the adequate and inadequate data sets by the curve $N^*(\lambda)$. The optimal tolerance can be located when the local minimum is clearly developed on the performance curves $\sigma_F^{\text{Test}}$ and $\sigma_F^{\text{Train}}$. For noiseless data, $\epsilon^*$ is equal to zero and $\sigma_F^{\text{Test}}(N, 0, 0)$ converges to but is greater than $\sigma_F^{\text{Train}}(N, 0, 0)$ for any value of $N$. Therefore, $\epsilon^*$ can only be located for a noiseless data set of infinite size. The adequate amount of noiseless data $N^*(0)$ is equal to infinity, as shown in Figs. 5.1 and 5.2. In other words, since there is no noise, no local feature of the actual mapping is destroyed and consequently, the amount of data needed to perfectly represent the evident, local features is equal to infinity. The existence of noise in measurements significantly increases the amount of data needed to approximate the actual mapping to some details. Hence, for large amplitude of noise, the adequate amount of data $N^*(\lambda)$ is an increasing function of $\lambda$. Since the curve $N^*(\lambda)$ is equal to infinity for $\lambda = 0$ and $\lambda = +\infty$, it should have at least one minimum in the interval $[0, +\infty]$. We have shown the location of this minimum in Figs. 5.1 and 5.2 by $\lambda^*$.

The existence of a local minimum in the curve $N^*(\lambda)$ can also be explained as follows: $N^*$ represents the amount of data required to represent the main, local features of the actual mapping with a good confidence. Good confidence means enough data to fit each local feature with locally supported parametric approximations that reach a good confidence level for values of their parameters. As we also mentioned for $N_{\text{min}}$, for a constant level of confidence more data are required as the amplitude of noise increases. Hence for a fixed number of main, local features, more data are needed to adequately represent these features as the noise amplitude increases. The confidence requirement for an adequate data set makes its size $N^*$ be an increasing function of the amplitude of noise. On the other hand, reducing the amount of noise increase the number of local features that have not been destroyed by noise. In other words, the number of main, local features that are evident is a decreasing function of the noise amplitude. Noise destroys features that are more local and more global features survive. This characteristic makes the illusion that the mapping underlying a noisy data set is smoother than the actual mapping. Since (1) the adequate amount of data $N^*(\lambda)$ is the sum of amounts of data required to adequately represent all main, local features that are evident, (2) the adequate amount of data for representing a local feature is an increasing function of the noise amplitude, and (3) the number of evident, main, local features is a decreasing function of the amplitude of noise, $N^*(\lambda)$ generally has a local minimum.

Recalling the definition for the measure of approximation confidence discussed in Section 5.3, the curve $N^*(\lambda)$ represents the minimum amount of data required to be able to compute the approximation confidence measure for an MC-HARP approximation. This property of $N^*(\lambda)$ should not lead to the conclusion that MC-HARP approximations built using data sets of size $N^*(\lambda)$ have the same confidence or accuracy. It just indicates that for data sets of size $N^*(\lambda)$ or larger, the approximation confidence measure for a constructed MC-HARP approximation is measurable. The approximation risk $\text{RISK}_{P(\epsilon^*)}$ is an increasing function of the noise amplitude $\lambda$ and is a decreasing function of the amount of data $N$. The contour curves of the accuracy index, measured by the approximation risk $\text{RISK}_{P(\epsilon^*)}$, on the quality-quantity plane ($N-\lambda$ plane) are increasing.
function of the noise amplitude. Hence by moving along the curve $N^*(\lambda)$ to go from a small amount of noise to a large amount of noise, the accuracy of a constructed MC-HARP approximation decreases. An MC-HARP approximation built using an adequate set of data with some noise amplitude, has better approximation accuracy and confidence than an MC-HARP approximation built using an adequate set of data with a larger noise amplitude.

We use the noise amplitude $\lambda^*$, indicating the location of the minimum of $N^*(\lambda)$, as a threshold between small amounts of noise and large amounts of noise. The line $\lambda = \lambda^*$, schematically shown on Figs. 5.1 and 5.2, is the fuzzy boundary between data sets with small amplitudes of noise and those with large amplitudes of noise. In previous sections we use qualitative measures of noise such as small or large without a definite definition of them. Now it is time to do so. We say the amplitude of noise added to a data set is small if it is smaller than $\lambda^*$, otherwise it is large.

$$\lambda < \lambda^* \iff \text{small amplitude of noise}$$
$$\lambda > \lambda^* \iff \text{large amplitude of noise}$$

(5.13)

A practical question about the proposed rating of the quality of data is how to compute $\lambda^*$ for real-world problems. We may answer: by plotting $N^*(\lambda)$. But this approach is not practical because for real-world problems, unlike our numerical simulations, the experimenter does not have the luxury of playing with the quantity and quality of data. Here based on our numerical simulations and the characteristics of MC-HARP, we give some hints to rate the quality of a given data set using the MC-HARP computable performance curves. These hints help an experimenter not to compute $\lambda^*$ but to qualitatively estimate the amplitude of noise. In other words, by using these hints, the experimenter will be able to conclude that the amount of noise in the given data is small or large. These hints are as follows:

**Hint 1:** The main difference between performance curves for small noise amplitudes and large noise amplitudes is the location of the primary peak of the $\sigma^\text{Train}_F$ curve that is represented by the critical point $\varepsilon_2$. For data sets with a small amount of noise, the primary peak is closer to $\varepsilon_{\text{max}}$ and the $\sigma^\text{Train}_F$ curve is skewed toward $\varepsilon_{\text{max}}$; i.e., $\varepsilon_2 > \varepsilon_{\text{max}}/2$. Conversely, when the amplitude of noise is large, $\varepsilon_2$ is closer to zero than to $\varepsilon_{\text{max}}$, as shown in Figs. 5.1 and 5.2. This hint can be summarized as follows

$$\varepsilon_2 > \frac{\varepsilon_{\text{max}}}{2} \iff \lambda < \lambda^*$$

(5.14)

When the amplitude of noise is large, an MC-HARP approximation with small subdomains associated with a small tolerance value follows the behavior of noisy data and the deviation among outputs predicted by its corresponding HARP approximations is dominated by the amplitude of noise. Consequently, for large noise amplitudes, the maximum value of the $\sigma^\text{Train}_F$ curve should associate with complex MC-HARP approximations. This characteristic explains why $\varepsilon_2$ is closer to zero than $\varepsilon_{\text{max}}$ when the amount of noise is large. On the other hand for the data sets with small amplitudes of noise, a complex MC-HARP approximation with small subdomains has
smaller deviation than a simple MC-HARP approximation because its corresponding HARP approximations have the flexibility to be close to the actual mapping and consequently to one another. Therefore, when the noise amplitude is small, the maximum values of the $\sigma_{\text{Train}}^2$ curve associates with simple MC-HARP approximations. In this case, $\varepsilon_2$ is closer to $\varepsilon_{\text{max}}$ than to zero.

**Hint 2:** If there is a premature local minimum on the performance curve $\sigma_{\text{Test}}^2$ at the left side of the primary peak of the performance curve $\sigma_{\text{Train}}^2$, then the amplitude of noise is small. This hint can be expressed as follows

$$\exists \varepsilon_5: \varepsilon_5 < \varepsilon_2 \Leftrightarrow \lambda < \lambda^*$$  \hspace{1cm} (5.15)

**Hint 3:** When $\varepsilon_{\text{max}}$ is much larger than the mean absolute value of the actual target values, the noise amplitude is large. The mean absolute value of the actual target values can be approximated by the mean absolute value of the noisy target values for training data points.

**Hint 4:** When the optimal tolerance $\varepsilon^*$ cannot be located, if by increasing the amount of data, the values of $\varepsilon_0$, $\varepsilon_1$, and/or the initial value of the performance curve $\sigma_{\text{Test}}^2$ at $\varepsilon=0$, $\sigma_{\text{Test}}^2(N, 0, \lambda)$, do not significantly vary, then the amplitude of noise is large.

**Hint 5:** When the secondary peak at $\varepsilon=\varepsilon_4$ or the local minimum at $\varepsilon=\varepsilon^*$ of the performance curve $\sigma_{\text{Train}}^2$ are at the left side of its primary peak at $\varepsilon=\varepsilon_2$, the amplitude of noise is small. This hint takes the form as follows

$$(\varepsilon_4 < \varepsilon_2 \text{ or } \varepsilon^* < \varepsilon_2) \Leftrightarrow \lambda < \lambda^*$$  \hspace{1cm} (5.16)

These hints are reliable only when the amount of data is larger than the minimal amount of data $N_{\text{min}}$. The experimenter can check this condition by verifying that the values and locations of maximum points of the performance curves $\text{RMS}_{\text{Train}}^2$ and $\sigma_{\text{Train}}^2$ do not vary significantly by changing the amount of data.

**Remark.** In this section we represent a data set by its number of data points $N$ and its amplitude of noise $\lambda$. Another main characteristic of a data set is the distribution of its data points in the input domain. Two data sets with the same amount of data points but different distributions develop two MC-HARP approximations with different performances. The observed characteristics for the performance of MC-HARP are based on a uniform distribution of data points. For a uniformly distributed set of data points, the number of data points $N$ is the only parameter that controls the density (quantity) of data points. The concluded results about MC-HARP can be generalized for nonuniform distributions of data points by including the distribution of data points in the definitions of the performance indices defined in Section 4.4.

### 5.8 MC-HARP Framework for Classifying Nonparametric, Data-fitting Problems

In Section 5.1, we discussed the need for establishing a framework for classifying the condition of a given data set. The proposed framework should help an experimenter to qualitatively and quantitatively examine
the data and the constructed approximation. The framework should be able to detect ill-conditioned data fitting problems and to warn the experimenter about the inadequacy of available data and the unreliable performance of the mathematical model constructed from the given data. In previous sections, we have shown that MC-HARP can establish measures for the quality and quantity of a data set. Here we present the MC-HARP proposed framework for classifying nonparametric, data-fitting problems with respect to their conditioning. (From now on, both terms: the conditioning of a data-fitting problem and the conditioning of a data set represent the same concept).

A data set can be represented by the noise amplitude \( \lambda \) and the number of data points \( N \) when the distributions of noise and data points are known. A given data set is a point on the quality-quantity plane \((N-\lambda)\) plane. As shown in Fig. 5.5, the curves \( N_{\text{min}}(\lambda) \) and \( N^*(\lambda) \) and the line \( \lambda = \lambda^* \) partition the \( N-\lambda \) plane into five regions. These quality-quantity regions are named \( \mathcal{QQ}_0 \) through \( \mathcal{QQ}_4 \). These regions are also shown in Figs. 5.1 and 5.2 for our numerical simulations. Before defining these quality-quantity regions, we again emphasize that the boundaries between these regions are fuzzy and the conditioning of a data set is a continuous entity that means by crossing the boundary between two quality-quantity regions, the conditioning of a data set does not significantly change.

The \( \mathcal{QQ}_o \) region includes all data sets that do not contain the minimal amount of data points. A data-fitting problem whose data set belongs to the \( \mathcal{QQ}_o \) region is ill-conditioned and its solution does not reliably capture the main features of the actual mapping and has poor confidence. The amount of data points in a data set that belongs to the \( \mathcal{QQ}_o \) region is smaller than \( N_{\text{min}} \). When a data-fitting problem is ill-conditioned \((i.e., N<N_{\text{min}})\), the values of \( \varepsilon_{\max} \). \( \text{RMS}_{\text{F}}^{\text{Train}}(N, \varepsilon_{\max}, \lambda) \). \( \sigma_{\text{F}}^{\text{Train}}(N, \varepsilon_2, \lambda) \), and \( \varepsilon_2 \) significantly change by increasing the amount of data.

A data-based approximation problem with a data set containing more data points than \( N_{\text{min}} \) is not ill-conditioned but it may not be significantly well-conditioned. The true measure for the conditioning of a data set is the approximation accuracy of its corresponding MC-HARP approximation, \( \text{RISK}_{\text{F}(\varepsilon*)} \). The lower the approximation risk \( \text{RISK}_{\text{F}(\varepsilon*)} \) is, the better the conditioning of the data set is. The approximation accuracy
RISK_{F(e^*)} increases by increasing the amount of data and by decreasing the amount of noise. For a fixed amplitude of noise, a data set with a large amount of data has better conditioning than a data set with a small amount of data. Furthermore, for a fixed amount of data, a data set with a small amplitude of noise has better conditioning than a data set with a large amplitude of noise.

The problem with using RISK_{F(e^*)} as a measure for the conditioning of a data set is that the approximation risk needs values of the actual mapping and consequently is not computable for actual data sets. As we proposed in Section 5.4, the confidence measure \( \sigma_{F(e^*)}^{\text{Test}} \) is a reasonable approximation for the approximation accuracy. The approximation confidence is measurable when the amount of data is adequate; i.e., \( N \geq N^* \). Hence we can classify data sets with respect to the measurability of their conditioning. We say a data set and its corresponding data-fitting problem are *measurably-conditioned* when the approximation confidence of its corresponding MC-HARP approximation is measurable. A measurably-conditioned data set contains an adequate amount of data, \( N \geq N^* \). The approximation confidence measure \( \sigma_{F(e^*)}^{\text{Test}} \) is also a quantitative measure for the conditioning of a data set. The lower the \( \sigma_{F(e^*)}^{\text{Test}} \) measure is, the better the conditioning of the data set is. The measurability of a data set does not infer the good conditioning of the data, but it does indicate that the computable confidence measure is a reasonable approximation for the conditioning of the data set. If the confidence measure is measurable and its value is acceptable, then the data-fitting problem and its data set have acceptable conditioning.

In the proposed MC-HARP framework for classifying nonparametric, data-fitting problems, the quality-quantity plane is partitioned by two boundaries: The \( N_{\text{min}}(\lambda) \) boundary separates ill-conditioned problems and the \( N^*(\lambda) \) boundary separates measurably-conditioned problems from the rest of data-fitting problems, respectively. As is shown in Table 5.1, the quality-quantity regions can be represented in a matrix. The regions \( \mathcal{Q}_1 \) and \( \mathcal{Q}_3 \) are the transitional zone between ill-conditioned data sets and the measurably-conditioned ones. Data sets in regions \( \mathcal{Q}_2 \) and \( \mathcal{Q}_4 \) have adequate amounts of data points and their conditionings are measurable. For a fixed amount of data, since a data set in \( \mathcal{Q}_2 \) has a smaller amount of noise than a data set in \( \mathcal{Q}_4 \), it has better conditioning and can develop a more accurate, approximated mapping. Furthermore, for a fixed amplitude of noise, since a data set in \( \mathcal{Q}_2 \) contains more data points than a data set in \( \mathcal{Q}_1 \), it has better conditioning. A similar property is true for \( \mathcal{Q}_4 \) in comparison with \( \mathcal{Q}_3 \).

In the proposed MC-HARP framework, a noise-free data set is not measurably-conditioned and belongs to the \( \mathcal{Q}_1 \) region when it is not ill-conditioned. This characteristic means that we cannot define a reasonable measure for the approximation accuracy of an MC-HARP approximation using noise-free data but it does not

<table>
<thead>
<tr>
<th>( N &lt; N_{\text{min}} )</th>
<th>( N_{\text{min}} \leq N &lt; N^* )</th>
<th>( N^* \leq N )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda \leq \lambda^* )</td>
<td>( \mathcal{Q}_2 )</td>
<td>( \mathcal{Q}_1 )</td>
</tr>
<tr>
<td>( \lambda &gt; \lambda^* )</td>
<td>( \mathcal{Q}_3 )</td>
<td>( \mathcal{Q}_4 )</td>
</tr>
</tbody>
</table>

Table 5.1 Classification of data sets

131
mean that the accuracy in approximation and consequently the conditioning for a noise-free data set is low. In the next section, we will present a series of rules for selecting the most suitable tolerance value named $\epsilon_s$ for data sets that are not ill-conditioned. The tolerance value $\epsilon_s$ is equal to the optimal tolerance value $\epsilon^*$ when the amount of data is adequate and is the best candidate for $\epsilon^*$ when the data set belongs to the regions $QQ_1$ and $QQ_3$. The deviation measure $\alpha \text{Test}_{F_{C(e)}}$ is an approximation for the conditioning measure of data sets in regions $QQ_1$ and $QQ_3$ including noise-free data sets.

For the real-world problems, an experimenter cannot play with the amounts of data and noise in order to plot the complete quality-quantity map, as shown in Fig. 5.5, for his problem. Our proposed framework should be accompanied by a series of rules for helping an experimenter to classify his data-fitting problem using only the MC-HARP computable performance curves. These rules should help an experimenter to know to which quality-quantity regions his data set belongs and what the conditioning of his data-fitting problem is. These rules for a given data set $\mathcal{D}$ containing $N$ data points with $\lambda$ amplitude of noise are represented by a decision tree in Fig. 5.6. In this decision tree for classifying a data set, the three main questions are: (1) Is the amount of data larger than the minimal amount of data; i.e., $N \geq N_{\text{min}}$? (2) Is the amount of noise small; i.e., $\lambda \leq \lambda^*$? and (3) Is the amount of data adequate; i.e., $N \geq N^*$? The answer to the first question is no if the values of $\epsilon_{\text{max}}, \text{RMS}_{F}^{\text{Train}}(N, \epsilon_{\text{max}}, \lambda), \sigma_{F}^{\text{Train}}(N, \epsilon_{2}, \lambda),$ and $\epsilon_{2}$ significantly change by increasing the amount of data. The answer to the second question can be found by using hints given in Section 5.7. Finally, the answer to the third question is yes if the optimal tolerance $\epsilon^*$ can be located on the performance curves $\sigma_{F}^{\text{Train}}$ and $\sigma_{F}^{\text{Test}}$.

Table 5.2 shows features that needed to be observed in the MC-HARP computable performance curves in order to assign a given data set to a specific quality-quantity region. Table 5.2 is useful when we want to check the adequacy of data. In other words, we want to know the given data set belongs to $QQ_1$ or $QQ_2$ when the amplitude of noise is small or it belongs to $QQ_3$ or $QQ_4$ when the amplitude of noise is large. The feature $s_1$ represents the requirement for the data set not to be ill-conditioned and the feature $s_6$ is the main requirement for the adequacy of a data set.

MC-HARP establishes a framework for classifying nonparametric, data-fitting problems with respect to the quality-quantity conditioning of their data sets. The MC-HARP framework proposes a quality-quantity map for a data set and a quantitative measure for its conditioning. Furthermore, the proposed framework in-
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>The values and locations of maximum points of performance curves $RMS_{Trin}$ and $\sigma_{Trin}$ do not significantly vary by increasing the amount of data $N$.</td>
</tr>
<tr>
<td>$s_2$</td>
<td>The tolerance values $\varepsilon_1$ and $\varepsilon_2$ do not significantly vary by increasing $N$.</td>
</tr>
<tr>
<td>$s_3$</td>
<td>The initial value of the performance curve $\sigma_f^{Trin}, \sigma_f^{Test}(N, 0, \lambda)$, does not significantly vary by increasing $N$.</td>
</tr>
<tr>
<td>$s_4$</td>
<td>The critical point $\varepsilon_5$ can be located.</td>
</tr>
<tr>
<td>$s_5$</td>
<td>The tolerance value $\varepsilon_3$ does not significantly vary by increasing $N$.</td>
</tr>
<tr>
<td>$s_6$</td>
<td>The optimal tolerance $\varepsilon^*$ can be located on performance curves $\sigma_f^{Trin}$ and $\sigma_f^{Test}$.</td>
</tr>
<tr>
<td>$s_7$</td>
<td>The critical point $\varepsilon_4$ can be located.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$N_{\text{min}} \leq N &lt; N^*$</th>
<th>$N^* \leq N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda \leq \lambda^*$</td>
<td>$s_1$, $s_4$ besides $s_1$, $s_4$ also $s_2$, $s_3$, $s_5$, $s_6$, $s_7$</td>
</tr>
<tr>
<td>$\lambda &gt; \lambda^*$</td>
<td>$s_1$, $s_2$, $s_3$ besides $s_1$, $s_2$, $s_3$ also $s_5$, $s_6$, $s_7$</td>
</tr>
</tbody>
</table>

Table 5.2 (a) Features in the MC-HARP computable performance curves, (b) Classification of a data set with respect to adequacy.

Incluces a series of rules based on the features in the MC-HARP computable performance curves in order to classify a data set with respect to its conditioning.

**Remark.** The proposed MC-HARP framework for classifying data sets is for nonparametric approximation. When the actual mapping $G$ belongs to the family $\Theta$ of parametric, subdomain approximations, the data-fitting problem is parametric. In this case, as mentioned in Section 5.6.3, MC-HARP can be used to verify that the data-fitting problem is parametric. However, since when $G$ belongs to $\Theta$, the optimal tolerance $\varepsilon^*$ is equal to $\varepsilon_{\text{max}}$ and $\sigma_f^{\text{Test}}(N, \varepsilon_{\text{max}}, \lambda)$ is equal to zero, the deviation measure for the optimal tolerance is not a measure for the conditioning of a data set unlike the nonparametric problems. For parametric problems, one can use well known statistical methods including cross-validation and bootstrap to examine the adequacy of data and measure the confidence in the constructed approximation.

5.9 Complexity Selection for an MC-HARP Approximation

In Section 5.2, we proposed using the deviation measure $\sigma_f$ as a model selection criterion. Furthermore, we have shown that the optimal complexity of an MC-HARP approximation is associated with the optimal tolerance value $\varepsilon^*$ that can be located on the performance curves $\sigma_f^{\text{Trin}}$ and $\sigma_f^{\text{Test}}$ by a local minimum when the amount of data is adequate. The proposed MC-HARP model selection method requires an adequate amount of data in order to select the optimal complexity, to compute the confidence in approximation, to compute confidence bounds for the accuracy in approximation, and to measure the conditioning of the data. The pro-
posed MC-HARP model selection method for nonparametric approximations prevents building an unreliable, approximate mapping using an insufficient amount of data. On the other hand, as we have mentioned in Section 5.7, a data set containing more data than \( N_{\text{min}} \) can represent some global features of the actual mapping and it represents all main, local features of the actual mapping when it contains an adequate amount of data. It seems that for the data sets containing more data points than \( N_{\text{min}} \) and less than \( N^* \), the optimal complexity \( e^* \) cannot be pinpointed like adequate data sets, but since these data sets contain some features of the actual mapping, we may be able to determine an interval to bound \( e^* \) and even select a reasonable candidate for \( e^* \).

In this section, we present a series of rules for bounding \( e^* \) and selecting the most suitable tolerance value named \( e_s \) for data sets that are not ill-conditioned. By increasing the amount of data from \( N_{\text{min}} \) to \( N^* \), the proposed bounds for \( e^* \) become more reliable and \( e_s \) becomes closer to \( e^* \). Using MC-HARP quality-quantity map proposed in Section 5.8, the proposed rules for selecting \( e_s \) and the bounds for \( e^* \) are for data sets in the \( QQ_1 \) and \( QQ_3 \) quality-quantity regions. For the \( QQ_2 \) and \( QQ_4 \) regions, the amount of data is adequate and \( e_s \) is equal to the optimal tolerance value \( e^* \) that can be located. For the \( QQ_0 \) region, the data-fitting problem is ill-conditioned, the optimal complexity \( e^* \) cannot be bounded, and the performance of an MC-HARP approximation regardless of its complexity is unreliable.

The main reasoning behind selecting \( e_s \) is that the constructed approximation should be simple when the amount of noise is large and is more complex when the amount of noise is small. Hence, for that data sets in the \( QQ_1 \) region, the selected tolerance \( e_s \) is closer to zero and for the data sets in \( QQ_3 \), \( e_s \) is closer to \( e_{\text{max}} \). The proposed rules for selecting \( e_s \) and bounding \( e^* \) for the \( QQ_1 \) region are as follows:

1. If the critical point \( e_3 \) can be located \underline{and} \( e_3 \) is less than \( e_2 \), then
   \[ e^* \text{ is bounded from above by } e_3 \text{ and from below by } e_0 \text{ if it can be located or by zero when } e_0 \text{ cannot be located.} \]

2. If there is a distinguished inflection point or a saddle region on the performance curves \( \sigma^\text{Train}_F \) and \( \sigma^\text{Test}_F \) that is close to \( e_3 \) then
   \[ e_s \text{ is represented by the inflection point or the lower bound of the saddle region. (Setting } e_s \text{ to be equal to } e_3 \text{ is also a reasonable choice).} \]

3. else
   If the critical point \( e_5 \) can be located then
   set \( e_s \) to be equal to \( e_5 \).
   else
   If the critical point \( e_6 \) can be located then
   setting \( e_s \) to be equal to \( e_3 \) or \( (e_6+e_3)/2 \) is reasonable.
   else
   setting \( e_s \) to be equal to \( e_3 \) or \( e_3/2 \) is reasonable.
endif
endif
else
If the critical point $\varepsilon_o$ can be located then

$$\varepsilon_o \leq \varepsilon^* < \varepsilon_2$$

and setting $\varepsilon_s$ to be equal to $\varepsilon_o$ prevents building an approximation with unreasonable complexity.

else

$\varepsilon^*$ cannot be bounded and $\varepsilon_s$ cannot be selected.

endif

For the data sets in the $QQ_2$ region, the optimal complexity $\varepsilon^*$ is definitely less than $\varepsilon_2$. The proposed rules for selecting $\varepsilon_s$ and bounding $\varepsilon^*$ for data sets in the $QQ_3$ region are as follows:

If the critical point $\varepsilon_3$ can be located and $\varepsilon_3$ is greater than $\varepsilon_2$, then

If there is a distinguished inflection point or a saddle region on the performance curves $\sigma_F^{Train}$ and $\sigma_F^{Test}$ that is close to $\varepsilon_3$ then

$\varepsilon_s$ is represented by the inflection point or the upper bound of the saddle region. (Setting $\varepsilon_s$ to be equal to $\varepsilon_3$ is also a reasonable choice). $\varepsilon_3 \leq \varepsilon^* < \varepsilon_{max}$.

else

$\varepsilon_3 \leq \varepsilon^* < \varepsilon_{max}$ and setting $\varepsilon_s$ to be equal to $\varepsilon_3$ is reasonable.

endif

else

$max(\varepsilon_o, \varepsilon_1, \varepsilon_2) \leq \varepsilon^* < \varepsilon_{max}$ and setting $\varepsilon_s$ to be equal to $max(\varepsilon_o, \varepsilon_1, \varepsilon_2)$ prevents building an approximation with unreasonable complexity.

endif

For data sets in the $QQ_4$ region, the optimal complexity $\varepsilon^*$ is definitely greater than $\varepsilon_2$. When the critical point $\varepsilon_3$ cannot be located, although we recommend to dismiss using a data-based, approximate mapping, the model selection techniques including resampling and penalizing techniques discussed in Section 4.2 can be used to select the complexity of an MC-HARP approximation.

Remark. The root predicted squared error $RPS$ is one of the penalizing techniques for model selection. In Figs. 4.2 through 4.5, we have shown the $RPS_F$ performance curve. The location of the minimum of the $RPS_F$ curve named $\varepsilon_p$ represents the optimal complexity with respect to the root-predicted-squared-error criterion. It is evident that although we used the exact value of the noise variance (that is usually unknown), $\varepsilon_p$ is only close to $\varepsilon^*$ when the amount of noise is small. For the large amplitude of noise, $\varepsilon_p$ is smaller than $\varepsilon^*$. Indicating that the $RPS_F$ criterion has the tendency to select overly complex approximations. However since the root predicted squared error can be easily computed for a constructed MC-HARP approximation, providing an estimate for the noise variance as discussed in Section 4.2, the $RPS_F$ performance curve can be used to prevent building MC-HARP approximations with unreasonable complexity. Furthermore, when $\varepsilon^*$ can be located by using the MC-HARP model selection technique, one can adjust the estimate for the noise variance such that $\varepsilon_p$ becomes equal to $\varepsilon^*$ and consequently computes an estimate for the noise amplitude.
5.10 Performance Study of MC-HARP Using Real Data

In this section, we apply the proposed MC-HARP model selection technique and framework for classifying data sets to build a mathematical model for a set of real data. The data set was provided as a part of building data analysis and prediction competition organized by the American Society of Heating, Refrigeration, and Air conditioning Engineers (ASHRAE) during the spring of 1993. Prediction using empirical models was the goal of the competition.

The data set consists of solar radiation measurements made by four fixed pyranometer devices used to predict the time varying hourly beam radiation during a 300-day period. This four-pyranometer device is used in an adaptive controller to predict building cooling loads. The data consist of four input (independent) variables and one output (dependent) variable. The input variables are solar fluxes measured by four pyranometers at hourly intervals during daylight hours. The input variables are: horizontal, tilted surface southeast, tilted surface south, and tilted surface southwest solar fluxes named $x_1$ through $x_4$, respectively. The output variable is the beam insolation. All variables are measured in watts per square meter. The full data set include 3344 measurements. A random sample of data from the full data set has been extracted to serve as the training set of 2444 data points; i.e., $N=2444$. The remaining 900 data points are test data points; i.e., $N_t=900$.

MC-HARP approximations with a linear subdomain approximation $\theta$ are used to build empirical models. A term truncation scheme is used to downsize $\theta$ for small subdomains. The least squares estimator is used to compute the parameters of $\theta$ during the subdomain training process. A ramp squashing function with $\mu=0$, as described in Section 2.2.5, is composed on the subdomain approximation $\theta$. The maximum training residual is chosen to be the termination criterion for the training process. The MC-HARP sample size $p$ is set to be 50. For the subdomain partitioning process, we use the linear partitioning function, defined in Eqn. (2.34), with $r=2$. For selecting the splitting thresholds, we choose the fuzzy $\mathcal{P}_\ell$ scheme.

To illustrate the use of MC-HARP for variable selection, we build MC-HARP approximations for different subsets of the independent variables $x_1$ through $x_4$. We construct empirical models for subsets of size three or four selected from the four input variables. The constructed approximate mappings using $(x_1, x_2, x_3), (x_1, x_2, x_4), (x_1, x_3, x_4), (x_2, x_3, x_4),$ and $(x_1, x_2, x_3, x_4)$ as their input variables and are named $x_1x_2x_3$, $x_1x_2x_4$, $x_1x_3x_4$, $x_2x_3x_4$, and $x_1x_2x_3x_4$ mappings, respectively. By changing the set of input variables, we want to study the sensitivity of the constructed mapping with respect to its input variables and to select the best set of variables. The sensitivity analysis for variable selection is a crucial process for building data-based mappings using real data because some of the variables in the data set may not be as significant as the rest of the variables and deleting these insignificant variables decreases the dimensionality of the constructed mapping and increases its reliability.

Figure 5.7 shows the performance curves for five mappings built for the solar flux data. In Fig. 5.7, we also show the root-predicted-squared-error $RPS_F$ curve by setting the standard deviation of noise to be 20. It is evident that for all constructed mappings, the $\sigma_{F\text{Train}}$ curve is skewed to the right. This observation supports Hint (5.14) and indicates that the amplitude of noise is small; i.e., $\lambda<\lambda^*$. For none of constructed mappings can the optimal tolerance $\epsilon^*$ be located. In other words, no local minimum is developed on both $\sigma_{F\text{Train}}$ and $\sigma_{F\text{Test}}$ curves. This observation indicates that the amount of data is not adequate; i.e., $N<N^*$. Consequent-
Fig. 5.7 Performance curves
ly the conditioning of the data set and the confidence in approximation are not measurable. However by playing with the amount of training data, we have observed that the location and values of maximum points of performance curves $RMS_{Train}$ and $\sigma_{Train}$ do not significantly vary by changing the amount of data. This observation leads to the conclusion that the amount of data is greater than $N_{min}$. Since $\lambda$ is less than $\lambda^*$ and $N_{min} < N < N^*$ by following the decision tree in Fig. 5.6, it can be concluded that the given data set belongs to the $QQ_1$ quality-quantity region.

The prediction error $RMS_{Test}$ is the measure for the approximation accuracy of the constructed MC-HARP mappings. As shown in Fig. 5.7, the $x_2x_3x_4$ and $x_1x_2x_3x_4$ mappings have the lowest $RMS_{Test}^*$ values. The optimal tolerance value $\varepsilon^*$ is associated with the location of the minimum of the $RMS_{Test}^*$ curve. The $x_1x_2x_4$ mapping has the largest prediction error among the five constructed mappings and its $RMS_{Test}^*$ curve is does not have a minimum. This observation indicates that the missing variable $x_3$, that is the tilted surface south solar flux, in the $x_1x_2x_4$ model is the most significant input variable. The same conclusion can be made without using the expected outputs for test data points by comparing the $\sigma_{Test}$ curves of the five constructed models. The approximation deviation $\sigma_{Test}^*$ for the $x_1x_2x_4$ mapping is much larger than the other mappings.

Table 5.3 shows a summary of the MC-HARP proposed framework for studying the performance of constructed mappings. The optimal tolerance $\varepsilon^*$ cannot be located by a local minimum on the performance curves $\sigma_{Train}^*$ and $\sigma_{Test}^*$. Consequently the critical point $\varepsilon_4$, representing the location of the secondary peak in the $\sigma_{Train}^*$ curve, cannot be located. Except for the $x_1x_2x_4$ mapping, the critical point $\varepsilon_3$ can be located and its values is shown in Table 5.3. This observation is another reason to conclude that the $x_1x_2x_4$ model has poor performance. Both $x_2x_3x_4$ and $x_1x_2x_3x_4$ mappings develop a premature local minimum in their $\sigma_{Test}^*$ curve. This property supports Hint (5.15) and indicates the amount of noise is definitely small for the training sets of these two mappings. The location of the premature local minimum is represented by $\varepsilon_5$ whose value is 70.

Using the rules in Section 5.9 for the data sets in the quality-quantity region $QQ_1$, the bounds for the optimal tolerance value $\varepsilon^*$ and the selected value for $\varepsilon_5$ are shown in Table 5.3 for all constructed mappings. It is evident that the recommended bounds for $\varepsilon^*$ using the MC-HARP computable performance curves reasonably bound the optimal tolerance $\varepsilon^*$ represented by the local minimum of the $RMS_{Test}^*$ curve. Furthermore,

<table>
<thead>
<tr>
<th>mapping</th>
<th>$\varepsilon_0$</th>
<th>$\varepsilon_1$</th>
<th>$\varepsilon_2$</th>
<th>$\varepsilon_3$</th>
<th>$\varepsilon_4$</th>
<th>$\varepsilon_5$</th>
<th>bounds for $\varepsilon^*$</th>
<th>$\varepsilon_3$</th>
<th>$\sigma_{Test}^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1x_2x_4$</td>
<td>100</td>
<td>160</td>
<td>360</td>
<td>200</td>
<td>100</td>
<td>0</td>
<td>$100 &lt; \varepsilon^*$</td>
<td>100</td>
<td>110</td>
</tr>
<tr>
<td>$x_1x_3x_4$</td>
<td>30</td>
<td>90</td>
<td>360</td>
<td>240</td>
<td>30</td>
<td>$30 &lt; \varepsilon^* &lt; 240$</td>
<td>240</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>$x_1x_2x_3$</td>
<td>30</td>
<td>70</td>
<td>600</td>
<td>280</td>
<td>30</td>
<td>$30 &lt; \varepsilon^* &lt; 280$</td>
<td>280</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>$x_2x_3x_4$</td>
<td>30</td>
<td>360</td>
<td>180</td>
<td>30</td>
<td>$30 &lt; \varepsilon^* &lt; 180$</td>
<td>70</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_1x_2x_3x_4$</td>
<td>30</td>
<td>360</td>
<td>180</td>
<td>70</td>
<td>$30 &lt; \varepsilon^* &lt; 180$</td>
<td>70</td>
<td>15</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.3 MC-HARP Performance analysis for constructed mappings
the selected tolerance \( \varepsilon_5 \) is the most suitable candidate for \( \varepsilon^* \). The deviation measure \( \sigma_{F(c_{(c)})}^{Test} \) is an approximate value for the approximation confidence and accuracy. As shown in Table 5.3, the \( x_1x_2x_4 \) mappings has the largest \( \sigma_{F(c_{(c)})}^{Test} \) value and the \( x_2x_3x_4 \) and \( x_1x_2x_3x_4 \) mappings have the smallest \( \sigma_{F(c_{(c)})}^{Test} \) value among the constructed mappings. The five MC-HARP mappings are sorted with respect to the \( \sigma_{F(c_{(c)})}^{Test} \) measure in the same way as they are sorted using \( RMS_{F(c_{(c)})}^{Test} \) that is the true measure for accuracy. It is evident in Fig. 5.7 that for the \( x_2x_3x_4 \) and \( x_1x_2x_3x_4 \) mappings the deviation measure \( \sigma_{F(c_{(c)})}^{Test} \) is a good approximation for \( RMS_{F(c_{(c)})}^{Test} \). These two models are the only models that a premature local minimum can be located on their \( \sigma_{F(c_{(c)})}^{Test} \) curves and their \( \varepsilon_5 \) value is associated with their \( \varepsilon_5 \) value. For the \( x_1x_2x_3 \) and \( x_1x_3x_4 \) mappings, the selected complexity \( \varepsilon_i \) corresponds to the critical tolerance \( \varepsilon_3 \). For the poor \( x_1x_2x_4 \) model, neither the critical point \( \varepsilon_3 \) nor the critical point \( \varepsilon_5 \) can be located and its selected complexity corresponds to \( \varepsilon_0 \). These observations indicate that an MC-HARP approximation that the critical point \( \varepsilon_3 \) can be located on its performance curves has better accuracy than an MC-HARP approximation for which \( \varepsilon_3 \) cannot be located. Furthermore, the existence of \( \varepsilon_5 \) is a sign for better performance.

Figure 5.8 shows complexity curves for the constructed mappings. It is evident that complexity curves for the \( x_2x_3x_4 \) and \( x_1x_2x_3x_4 \) mappings are lower than the other mappings. This observation indicates that these two models have less complexity than others and consequently, more reliable performance. On the other hand, the \( x_1x_2x_4 \) model has a high complexity curve that supports its poor performance. By comparing the complexity curves in Fig. 5.8 with the complexity curves in Fig. 4.11, it is evident that the complexity curves for the \( x_1x_2x_4 \) and \( x_1x_3x_4 \) models that have negative convexity are associated with poor data in comparison with other constructed models. Since the same data set is used for all models, a performance that indicates the data set is poor actually represents a poor performance. Furthermore for small tolerance values the number of parameters is much smaller than the number of data points, thus it can be concluded that the features in the data can be represented by the simple, linear subdomain approximation with good confidence. Consequently, the amount of data is definitely larger than the minimal amount of data \( N_{min} \).
Both performance and complexity analyses lead to the conclusion that the $x_2x_3x_4$ and $x_1x_2x_3x_4$ mappings models are the best constructed models. The selected complexity for these models is associated with the tolerance value equal to 70 corresponding to their $\varepsilon_5$ value. Since we know the optimal tolerance value from the MC-HARP framework for performance analysis, we adjust the estimate for the noise amplitude in the $RPS_F$ measure such that the local minimum of the $RPS_F$ curve is located at the optimal tolerance value, as shown in Fig. 5.7. The computed estimate for the noise amplitude is equal to 20, a value much smaller than the average of expected outputs for the training data points of 459. This observation supports the conclusion that the amount of noise is small.

Figure 5.9 shows the performance of the $x_2x_3x_4$ and $x_1x_2x_3x_4$ models associated with the tolerance value equal to 70. It is evident that both models have good performance for the training set. The $RMS_{Train}^F$ measure for the $x_2x_3x_4$ and $x_1x_2x_3x_4$ mappings is equal to 9.4 and 9.9, respectively. As shown in Figs. 5.9(c) and 5.9(d), the $x_2x_3x_4$ and $x_1x_2x_3x_4$ mappings have similar performance for the training and test set. This observation indicates that these two mappings have close predicted outputs for the entire input domain. The deviation measures $\sigma_F$ for the test data points are shown in Fig. 5.9(d). Both models have small deviation measures for the test set. The average deviation measure $\sigma_{Test}^F$ is about 15 for both models. Since the deviation measure $\sigma_{Test}^F$ is an approximation for the prediction error $RMS_{Test}^F$, both models should have good perfor-

140
Fig. 5.10 Performance of the best $x_1x_2x_3x_4$ mapping for the test set. Figure 5.10 shows the performance of the $x_1x_2x_3x_4$ model for the test set. The prediction error $RMS_{test}^F$ is 15.62 a value much smaller than the average of the expected outputs for the test data points of 351. Finally, the closeness of performances by the $x_2x_3x_4$ and $x_1x_2x_3x_4$ models indicates that the input variable $x_1$ representing the horizontal solar flux is the least significant, independent variable.

5.11 Conclusions

In this chapter, we have proposed to use the deviation measure $\sigma_F$ computed by MC-HARP as a model selection criterion. The optimal complexity of an MC-HARP approximation is associated with the optimal tolerance value $\varepsilon^*$ that can be located on the performance curves $\sigma_F^{Train}$ and $\sigma_F^{Test}$ by a local minimum when the amount of data is adequate. The proposed MC-HARP model selection is based on the minimization of the deviation measure $\sigma_F$ in the limit with respect to the amount of data over the entire input domain. The minimization in the limit and consideration of the entire input domain are the two main characteristics of the proposed model selection technique that distinguish it from current techniques which all use a straight minimization over the set of given data points or a subset of it.

We have expressed the MC-HARP philosophy for performance estimation of data-based approximate mappings. Unlike the sampling-based techniques for performance estimation, the MC-HARP method considers the spatial neighboring relations among data points and uses them not only for building its approximation but also for estimating a pointwise, approximation deviation measure. The MC-HARP philosophy for building an approximation and for estimating the performance of its approximation is compatible with the generalization goal of building data-based mathematical models. The MC-HARP method follows this generalization goal by building an approximation whose predicted output for a point in the input domain is strongly influenced by its neighboring data. The MC-HARP deviation measure represents how strong the nearby data for a point dictates its predicted output.

We have defined the deviation measure of an MC-HARP approximation for the optimal tolerance value $\varepsilon^*$ as a quantitative measure for the approximation confidence. The confidence in approximation is inversely related to the deviation in approximation $\sigma_F^{Test(\varepsilon^*)}$. The approximation confidence is measurable when the opti-
mal tolerance value can be located by a local minimum on the $\sigma_F^{\text{Train}}$ and $\sigma_F^{\text{Test}}$ performance curves. The deviation measure $\sigma_F^{\text{Test}_{\epsilon(e^*)}}$ is also a quantitative measure for the adequacy of data. Furthermore, we have shown that the computable confidence measure is a reasonable estimate for the approximation accuracy and can be used to compute confidence bounds for the approximation accuracy.

To represent the MC-HARP framework for classifying data-fitting problems, we have defined critical points $\epsilon_6$ through $\epsilon_5$ on the MC-HARP computable performance curves. We have explained the importance of these critical points and given hints to locate them. We have shown that these critical points can be used to verify that the actual mapping is representable by a presumed parametric function or to test the hypothesis that the given data is noiseless.

MC-HARP establishes a framework for classifying nonparametric, data-fitting problems with respect to the quality-quantity conditioning of their data sets. The MC-HARP framework proposes a quality-quantity map for a data set and a quantitative measure for its conditioning. The proposed framework defines a qualitative rating for the quality of data. The proposed rating is accompanied by a series of hints using the MC-HARP computable performance curves to help an experimenter to conclude that the amount of noise in the given data is small or large. Furthermore, the proposed framework is able to detect an ill-conditioned data-fitting problem and to warn the experimenter about the unreliable performance of the constructed mathematical model. Also the proposed MC-HARP framework introduce the concept of measurability for the conditioning of a data set. The computable confidence measure $\sigma_F^{\text{Test}_{\epsilon(e^*)}}$ is the quantitative measure for the conditioning of a data set. The proposed framework includes a series of rules based on the features in the MC-HARP computable performance curves to classify a data set with respect to its conditioning. Finally, the MC-HARP framework presents a series of rules for bounding the optimal tolerance value $\epsilon^*$ and selecting the most suitable tolerance value $\epsilon_5$ for a data set that is not ill-conditioned.

We have used a set of real data to illustrate the application of the proposed MC-HARP model selection technique and framework for classifying nonparametric, data-fitting problems. It has been shown that the MC-HARP method can select the most suitable complexity, estimate the amount of noise, approximate the prediction error, classify the given data set, and do variable selection. MC-HARP presents a complete package for building data-based, approximate mappings and analyzing performance of the constructed mathematical models.
CHAPTER SIX
MC-HARP Data-based Mathematical Modeling
for Long Term Pavement Performance

“Better is the enemy of the good.”
Voltaire

A pavement structure is a layered system designed to distribute traffic loads to the subgrade, as shown in Fig. 6.1. The condition of a pavement is influenced by its structural properties, traffic, and environmental effects. The structure of a pavement generally consists of three layers designated as the subbase, base, and surface. The structural behavior of a pavement is a function of the geometric and material properties of its three layers and the subgrade. Traffic is a variable load and has many parameters including, traffic volume and its history, vehicle weight, speed, tire pressure, axle spacing, and vehicle suspension. Environmental conditions including temperature, humidity, and precipitation vary in an uncontrolled fashion and inflict damage to the pavement. The factors affecting the behavior of a pavement system are coupled. For example, material properties of flexible pavements are influenced not only by temperature, but also by vehicle speed. Furthermore, for modeling a pavement system an objective measure is needed for the pavement performance. The amount and variability of factors affecting the behavior of a pavement make it an extremely complex system. Thus, it is tempting for the pavement engineer to make simplifying assumptions about those factors that significantly affect pavement response. The experiment for modeling a pavement system was conducted by the American Association of State Highway Officials (AASHO) during the period of 1958 through 1960. The AASHO Road Test has been the basis for pavement design practices during the last thirty years.

In this chapter we briefly described the AASHO Road Test and its performance data. We use expected trends for a reasonable model for pavement performance to develop a test set and criteria to measure the performance of any empirical model for the test set. Then we described the recommended AASHO empirical

![Pavement structure diagram]

Fig. 6.1 Pavement structure
model for pavement performance and study its performance for the training and test sets. Then we show the applicability of the MC-HARP method for building data-based mathematical models for the long term pavement performance. We build MC-HARP approximations with performance superior to the AASHO model. We use the subdomain approximation that inherently includes *a priori* knowledge about a model for pavement performance and select suitable complexity to improve the performance of an MC-HARP empirical model. Finally we use the results from MC-HARP analysis to verify the suitability of the AASHO model for different region of its input domain.

6.1 AASHO Road Test

The principal goal of the pavement research in the AASHO Road Test was to establish relationships showing how the performance of a pavement is influenced by its structural design, represented by layer thicknesses of the pavement structure, and loading, represented by the magnitude and frequency of axle loads, for both rigid and flexible pavements. Thus experimental designs and analytical procedures were developed to obtain the effects of controlled experimental factors. The Road Test did not make it possible to obtain effects for other factors that were either held constant or that varied in an uncontrolled fashion, for example, embankment soil, strength of materials, and environmental conditions. In the following we describe the factorial experiment designed during the Road Test to model how the pavement performance is influenced by its structural design and loading. Then we explain how the performance of a pavement system was measured during the Road Test and define the AASHO mathematical model for a pavement system.

At the AASHO Road Test site, six loops were constructed. Each loop was a segment of a four-lane divided highway whose parallel roadways (tangents) were connected by a turnaround at each end. The north tangent of each of the six loops was constructed of flexible pavements (asphaltic concrete surface) and the south tangents were constructed of rigid pavements (portland cement concrete surface). The centerline divided pavements into inner and outer lane. Each section of tangents had the same structural design but different axle load applied on two lanes. The six north tangents included a total of 234 structural sections or 468 test sections. A majority of the sections in each pavement tangent comprised a complete factorial experiment whose design factors were surface, base, and subbase thicknesses. These experiments were referred to as the main factorial designs.

Loop 1 was not subjected to test traffic and used for special load studies and observing the effect of environment on pavements not subjected to traffic. Loops 2 through 6 were subjected to traffic for slightly more than two years. Ten different vehicle types were used for the ten traffic lanes of five traffic loops. The vehicles differed from one another based on the load and type of their load axles (single or tandem). Only one vehicle type was used in each lane so that each pavement section was subjected to only one type of loading. The distribution of vehicles was such that axle load applications accumulated at the same rate in all traffic lanes throughout the test period. Whenever possible, the vehicles traveled at a constant speed of 35 mph. The traffic schedule was maintained except when road conditions were prohibitive because of pavement distress or weather conditions. A total of 1,114,000 axle load applications was achieved during the test period.

In the AASHO Road Test, 164 test sections in the traffic loops 2 through 6 were used for the main factorial experiments of flexible pavement. The design variables, as shown in Fig. 6.1 were the surface thickness $D_1$, 

144
base thickness $D_2$, and subbase thickness $D_3$. They took discrete values of \{1, 2, 3, 4, 5, 6\}, \{0, 3, 6, 9\}, and \{0, 4, 8, 12, 16\} inches, respectively. The axle load $L$ took on discrete values 2, 6, 12, 18, 22.4, and 30 kips for single-axle test traffic. Table 6.1 shows the values of pavement design variables $D_1$, $D_2$, $D_3$, and axle load $L$ for each test section in the main factorial experiments for flexible pavements. Considerable overlap was provided between factorial experiments of different loops. In each loop a certain number of the designs were replicated. Variation in the performance of replicated sections provided a measure of the effects of uncontrolled variables during the Road Test. There were 20 pairs of replicate sections for flexible pavements under single-axle traffic in the Road Test and are shown as shaded areas in Table 6.1.

In the Road Test, the present serviceability index (PSI) of each section was determined every two weeks. Each 2-week period was referred to as index period and its last day was called index day. There were 55 index days during the test period and they were numbered sequentially from one to 55. The serviceability history of a section was considered to be completed when its PSI fell under the terminal serviceability. The terminal PSI at the Road Test was set to be 1.5. The life period of a section was started after its construction and finished when its serviceability reached the selected terminal serviceability. Test sections with thick structural design or light axle load had life periods longer than the testing period. The sections with thin design or heavy axle load reached their terminal serviceability before the end of the Road Test.

The serviceability of a pavement is defined as the ability to serve high-speed, high-volume automobile and truck traffic. The concept of present serviceability was used to represent the momentary ability of a pavement to serve traffic. For the AASHO Road Test, the Present Serviceability Rating (PSR) procedure was developed for periodic rating of the serviceability of pavements. The PSR procedure consisted of the mean of individual ratings by a selected panel of adjudicators with long experience in all aspects of highway engineering, and as highway users. A scale of 0 to 5 was established for panel ratings, with a value of 5 for the ideal pavement. Although the present serviceability is a subjective matter, for the AASHO Road Test a regression equation was developed to relate objective measurements of longitudinal and transverse profile variations and the amount of cracking and patching to the subjective panel rating. The measure computed by the developed regression equation using the distress measurements was referred to as the Present Serviceability Index (PSI). The present serviceability index is the measured, or estimated level of serviceability at any time during the life of the pavement.

In the Road Test by running the same type of vehicle with a constant speed on a test section during the test period, the traffic load had only two variable but controlled parameters; the axle load and number of axle load applications. Each load application on any test section represented the same axle load during the Road Test. For each test section the history of load applications was defined by the variation of the number of accumulated applications computed for the index days belonging to the life period of the section. The number of load applications $n_i$ made during the $r$th index period was multiplied by a seasonal weight value $a_i$ to consider effects of environmental conditions in the rate of pavement damage accumulated with applications of axle load. The weighted accumulated axle load applications for the $r$th index day $W_r$ is computed as follows

$$W_r = \sum_{i=1}^{r} a_i n_i$$

(6.1)
<table>
<thead>
<tr>
<th>Section No</th>
<th>Thickness (in)</th>
<th>Axle load (kips)</th>
<th>Section No</th>
<th>Thickness (in)</th>
<th>Axle load (kips)</th>
<th>Section No</th>
<th>Thickness (in)</th>
<th>Axle load (kips)</th>
<th>Section No</th>
<th>Thickness (in)</th>
<th>Axle load (kips)</th>
</tr>
</thead>
<tbody>
<tr>
<td>721</td>
<td>1.0</td>
<td>0.0</td>
<td>746</td>
<td>3.0</td>
<td>3.0</td>
<td>6.0</td>
<td>6.0</td>
<td>8.0</td>
<td>18.0</td>
<td>4.0</td>
<td>9.0</td>
</tr>
<tr>
<td>727</td>
<td>1.0</td>
<td>0.0</td>
<td>750</td>
<td>3.0</td>
<td>6.0</td>
<td>0.0</td>
<td>6.0</td>
<td>6.0</td>
<td>18.0</td>
<td>5.0</td>
<td>3.0</td>
</tr>
<tr>
<td>743</td>
<td>1.0</td>
<td>3.0</td>
<td>764</td>
<td>3.0</td>
<td>6.0</td>
<td>4.0</td>
<td>6.0</td>
<td>8.0</td>
<td>18.0</td>
<td>4.0</td>
<td>0.0</td>
</tr>
<tr>
<td>755</td>
<td>1.0</td>
<td>6.0</td>
<td>165</td>
<td>2.0</td>
<td>0.0</td>
<td>0.0</td>
<td>12.0</td>
<td>6.0</td>
<td>8.0</td>
<td>5.0</td>
<td>3.0</td>
</tr>
<tr>
<td>759</td>
<td>1.0</td>
<td>6.0</td>
<td>125</td>
<td>0.0</td>
<td>0.0</td>
<td>4.0</td>
<td>12.0</td>
<td>6.0</td>
<td>0.0</td>
<td>5.0</td>
<td>6.0</td>
</tr>
<tr>
<td>771</td>
<td>1.0</td>
<td>6.0</td>
<td>152</td>
<td>2.0</td>
<td>0.0</td>
<td>8.0</td>
<td>12.0</td>
<td>6.0</td>
<td>4.0</td>
<td>8.0</td>
<td>4.0</td>
</tr>
<tr>
<td>779</td>
<td>5.0</td>
<td>3.0</td>
<td>185</td>
<td>2.0</td>
<td>0.0</td>
<td>8.0</td>
<td>12.0</td>
<td>6.0</td>
<td>4.0</td>
<td>8.0</td>
<td>4.0</td>
</tr>
<tr>
<td>771</td>
<td>5.0</td>
<td>3.0</td>
<td>133</td>
<td>0.0</td>
<td>3.0</td>
<td>8.0</td>
<td>12.0</td>
<td>6.0</td>
<td>3.0</td>
<td>8.0</td>
<td>4.0</td>
</tr>
<tr>
<td>797</td>
<td>0.0</td>
<td>0.0</td>
<td>113</td>
<td>2.0</td>
<td>3.0</td>
<td>0.0</td>
<td>12.0</td>
<td>5.0</td>
<td>8.0</td>
<td>4.0</td>
<td>3.0</td>
</tr>
<tr>
<td>799</td>
<td>0.0</td>
<td>0.0</td>
<td>3.0</td>
<td>0.0</td>
<td>4.0</td>
<td>12.0</td>
<td>5.0</td>
<td>0.0</td>
<td>12.0</td>
<td>5.0</td>
<td>0.0</td>
</tr>
<tr>
<td>813</td>
<td>0.0</td>
<td>0.0</td>
<td>159</td>
<td>2.0</td>
<td>3.0</td>
<td>0.0</td>
<td>8.0</td>
<td>12.0</td>
<td>5.0</td>
<td>6.0</td>
<td>4.0</td>
</tr>
<tr>
<td>819</td>
<td>0.0</td>
<td>0.0</td>
<td>132</td>
<td>2.0</td>
<td>3.0</td>
<td>0.0</td>
<td>8.0</td>
<td>12.0</td>
<td>5.0</td>
<td>6.0</td>
<td>4.0</td>
</tr>
<tr>
<td>821</td>
<td>0.0</td>
<td>0.0</td>
<td>137</td>
<td>2.0</td>
<td>3.0</td>
<td>0.0</td>
<td>8.0</td>
<td>12.0</td>
<td>5.0</td>
<td>6.0</td>
<td>4.0</td>
</tr>
<tr>
<td>833</td>
<td>0.0</td>
<td>0.0</td>
<td>163</td>
<td>2.0</td>
<td>3.0</td>
<td>0.0</td>
<td>8.0</td>
<td>12.0</td>
<td>5.0</td>
<td>6.0</td>
<td>4.0</td>
</tr>
<tr>
<td>839</td>
<td>0.0</td>
<td>0.0</td>
<td>199</td>
<td>2.0</td>
<td>3.0</td>
<td>0.0</td>
<td>8.0</td>
<td>12.0</td>
<td>5.0</td>
<td>6.0</td>
<td>4.0</td>
</tr>
<tr>
<td>855</td>
<td>0.0</td>
<td>0.0</td>
<td>147</td>
<td>2.0</td>
<td>3.0</td>
<td>0.0</td>
<td>8.0</td>
<td>12.0</td>
<td>5.0</td>
<td>6.0</td>
<td>4.0</td>
</tr>
<tr>
<td>869</td>
<td>0.0</td>
<td>0.0</td>
<td>147</td>
<td>2.0</td>
<td>3.0</td>
<td>0.0</td>
<td>8.0</td>
<td>12.0</td>
<td>5.0</td>
<td>6.0</td>
<td>4.0</td>
</tr>
<tr>
<td>879</td>
<td>0.0</td>
<td>0.0</td>
<td>147</td>
<td>2.0</td>
<td>3.0</td>
<td>0.0</td>
<td>8.0</td>
<td>12.0</td>
<td>5.0</td>
<td>6.0</td>
<td>4.0</td>
</tr>
<tr>
<td>893</td>
<td>0.0</td>
<td>0.0</td>
<td>147</td>
<td>2.0</td>
<td>3.0</td>
<td>0.0</td>
<td>8.0</td>
<td>12.0</td>
<td>5.0</td>
<td>6.0</td>
<td>4.0</td>
</tr>
<tr>
<td>903</td>
<td>0.0</td>
<td>0.0</td>
<td>147</td>
<td>2.0</td>
<td>3.0</td>
<td>0.0</td>
<td>8.0</td>
<td>12.0</td>
<td>5.0</td>
<td>6.0</td>
<td>4.0</td>
</tr>
<tr>
<td>903</td>
<td>0.0</td>
<td>0.0</td>
<td>147</td>
<td>2.0</td>
<td>3.0</td>
<td>0.0</td>
<td>8.0</td>
<td>12.0</td>
<td>5.0</td>
<td>6.0</td>
<td>4.0</td>
</tr>
<tr>
<td>903</td>
<td>0.0</td>
<td>0.0</td>
<td>147</td>
<td>2.0</td>
<td>3.0</td>
<td>0.0</td>
<td>8.0</td>
<td>12.0</td>
<td>5.0</td>
<td>6.0</td>
<td>4.0</td>
</tr>
<tr>
<td>903</td>
<td>0.0</td>
<td>0.0</td>
<td>147</td>
<td>2.0</td>
<td>3.0</td>
<td>0.0</td>
<td>8.0</td>
<td>12.0</td>
<td>5.0</td>
<td>6.0</td>
<td>4.0</td>
</tr>
<tr>
<td>903</td>
<td>0.0</td>
<td>0.0</td>
<td>147</td>
<td>2.0</td>
<td>3.0</td>
<td>0.0</td>
<td>8.0</td>
<td>12.0</td>
<td>5.0</td>
<td>6.0</td>
<td>4.0</td>
</tr>
<tr>
<td>903</td>
<td>0.0</td>
<td>0.0</td>
<td>147</td>
<td>2.0</td>
<td>3.0</td>
<td>0.0</td>
<td>8.0</td>
<td>12.0</td>
<td>5.0</td>
<td>6.0</td>
<td>4.0</td>
</tr>
<tr>
<td>903</td>
<td>0.0</td>
<td>0.0</td>
<td>147</td>
<td>2.0</td>
<td>3.0</td>
<td>0.0</td>
<td>8.0</td>
<td>12.0</td>
<td>5.0</td>
<td>6.0</td>
<td>4.0</td>
</tr>
<tr>
<td>903</td>
<td>0.0</td>
<td>0.0</td>
<td>147</td>
<td>2.0</td>
<td>3.0</td>
<td>0.0</td>
<td>8.0</td>
<td>12.0</td>
<td>5.0</td>
<td>6.0</td>
<td>4.0</td>
</tr>
<tr>
<td>903</td>
<td>0.0</td>
<td>0.0</td>
<td>147</td>
<td>2.0</td>
<td>3.0</td>
<td>0.0</td>
<td>8.0</td>
<td>12.0</td>
<td>5.0</td>
<td>6.0</td>
<td>4.0</td>
</tr>
<tr>
<td>903</td>
<td>0.0</td>
<td>0.0</td>
<td>147</td>
<td>2.0</td>
<td>3.0</td>
<td>0.0</td>
<td>8.0</td>
<td>12.0</td>
<td>5.0</td>
<td>6.0</td>
<td>4.0</td>
</tr>
<tr>
<td>903</td>
<td>0.0</td>
<td>0.0</td>
<td>147</td>
<td>2.0</td>
<td>3.0</td>
<td>0.0</td>
<td>8.0</td>
<td>12.0</td>
<td>5.0</td>
<td>6.0</td>
<td>4.0</td>
</tr>
<tr>
<td>903</td>
<td>0.0</td>
<td>0.0</td>
<td>147</td>
<td>2.0</td>
<td>3.0</td>
<td>0.0</td>
<td>8.0</td>
<td>12.0</td>
<td>5.0</td>
<td>6.0</td>
<td>4.0</td>
</tr>
</tbody>
</table>

Table 6.1 Designs for the factorial experiment of flexible pavements under single-axle traffic
During the Road Test, the seasonal effects were measured by monitoring the variation of the average deflection under a 6-kip wheel load of 8 sections in the untrafficked loop 1 for an index period. The seasonal weight value $a_t$ was the ratio of the deflection measured during the $t$th index period to the average of measured deflections for the test period. The adjusting value $a_t$ for environmental conditions was greater than one during the $t$th index period when the untrafficked pavement was weaker than normal (i.e., its measured deflection was less than its average deflection during the Road Test) and less than one when it was stronger than normal. It was observed that the seasonal weight factor was effective in modeling the performance of flexible pavements but was less pronounced for rigid pavements. This observation indicates that although the condition of rigid pavements is associated with environmental effects, the seasonal weight factor cannot model environmental effects for rigid pavements. Furthermore, for a flexible pavement the seasonal weight factor is not adequate for modeling all environmental effects on the pavement performance (we will discuss this observation in Section 6.4).

During the Road Test, for each test section the histories of its present serviceability and number of accumulated load applications were measured. For each section, the pavement performance was defined based on the variation of the present serviceability index $P$ with respect to the accumulated axle load applications $W$. The number of accumulated load applications $W$ could be weighted or unweighted. The decay curve of $P$ with respect to $W$ was called the serviceability trend. The serviceability trend of each section was smoothed using a moving average that included at least three (and generally five) successive index values. In Appendix A of the AASHO report (1962), five pairs of $P$ and $W$ were selected to represent the serviceability trend for each section. If for a test section its present serviceability index had fallen to 1.5 during the Road Test, the values of $W$ were represented for $P$ equal to 3.5, 3.0, 2.5, 2.0, and 1.5, otherwise, the pairs of $W$ and $P$ at the index days 11, 22, 33, 44, and 55 were selected.

The main goal of the Road Test was to determine relationships between the performance of the pavement and the pavement design variables subject to various loads. The serviceability trend represented the performance of a pavement during the Road Test. The main assumption made at the Road Test was that the loss of pavement serviceability is because of the composite effects of traffic (axle load and its number of applications) and pavement structure (layer thicknesses). In other words, the AASHO Road Test simplifies the mathematical model for a pavement system to be a 5-dimensional mapping as follows

$$P = F(D_1, D_2, D_3, L, W)$$

where $D_1, D_2,$ and $D_3$ are thicknesses of the surface, base, and subbase layers, respectively, $L$ is the axle load, $W$ is the accumulated axle load applications, and $P$ is the present serviceability index.

The functional form and parameters of the model defined in Eqn. (6.2) can be estimated using the data measured during the Road Test. As a result of the Road Test, a data-based mathematical model can be built for the long term performance of a pavement. Such a model can be used for the design and performance analysis of pavement structures. In the following sections we study the performance of the model that is developed in the AASHO report (1962) and has been the basis for pavement design practices during the last thirty years. Then we use the AASHO Road Test data to build an MC-HARP approximation for the model defined in Eqn.
(6.2) and compare its performance with the AASHO model. Finally we use the MC-HARP method to construct an MC-HARP approximation whose parametric subdomain approximation is the AASHO model. We show that MC-HARP can improve the performance of the AASHO model and represents the Road Test measurements more effectively.

6.2 AASHO Model for a Pavement System

A parametric approach is used in the AASHO report (1962) to build a mathematical model for the multivariate mapping defined in Eqn. (6.2). The functional form of the multivariate mapping $F$ is presumed and its parameters are estimated such that the constructed mathematical model represents the data measured in the Road Test in an acceptable way. The basis of the AASHO model is a decay curve, wherein it is assumed that the condition of a pavement will deteriorate with accumulated traffic. The mathematical model chosen in the AASHO report takes the form

$$P = P_0 - (P_0 - P_t) \left(\frac{W}{Q}\right)^\beta$$

(6.3)

where $W$ is the number of axle applications that will reduce the present serviceability index from the initial serviceability level $P_0$ to $P$, $Q$ represents the number of axle applications at the terminal serviceability $P_t$, and $\beta$ is a shape factor. For $\beta=1$, the serviceability trend is a linear function of the accumulated axle load applications; if $\beta>1$, the serviceability loss rate increases with load applications; and if $\beta<1$, the serviceability loss rate decreases with respect to $W$. The initial serviceability ($P_0$) is considered to be the serviceability of the freshly constructed, untrafficked pavement. The ideal pavement has to be rare. In fact, newly constructed flexible pavement at the Road Test reflected an average serviceability index value of 4.2. The terminal serviceability ($P_t$) is considered to be that level of serviceability at which the pavement is deemed to be no longer performing its required function and should be resurfaced or reconstructed. The terminal serviceability $P_t$ was set to be 1.5 at the Road Test. The traffic capacity of a pavement is represented by $Q$ in the AASHO model.

The functional form in Eqn. (6.3) states that the proportion of usable serviceability consumed (i.e., $(P_0 - P)/(P_0 - P_t)$) is represented by a power function of the cumulative traffic at any time as a proportion of the traffic capacity of the pavement. (i.e., $W/Q$). The serviceability loss rate controlled by $\beta$ and the traffic capacity represented by $Q$ are assumed to be functions of the pavement design and traffic. On this basis the functional relationships for $\beta$ and $Q$ take the form as

$$Q = \frac{A_o(D + a_2)^{A_1}}{(L + 1)^{A_2}}$$

(6.4)

$$\beta = \beta_o + \frac{B_o(L + 1)^{B_2}}{(D + a_4)^{B_1}}$$

(6.5)

$$D = a_1D_1 + a_2D_2 + a_3D_3$$

(6.6)
The same functional form has been used for the performance of rigid pavements. In Eqns. (6.4) through (6.6), the axle load \( L \) and design thicknesses \( D_1', D_2', \) and \( D_3' \) are known for each test section of the AASHO Road Test. The unknown coefficients \( A_{0-2}, B_{0-2}, \beta_{0}, \) and the layer coefficients \( a_{1-4} \) were estimated through a regression analysis using the performance data of the AASHO Road Test. The developed model for the performance of flexible pavements subjected to single-axle traffic is

\[
P = 4.2 - 2.7(W/Q)\beta
\]  \(6.7\)

where

\[
\beta = 0.4 + \frac{0.081 (L + 1)^{3.23}}{(D + 1)^{5.19}}
\]  \(6.8\)

\[
Q = \frac{10^{5.93} (D + 1)^{9.36}}{(L + 1)^{4.79}}
\]  \(6.9\)

\[
D = 0.44D_1 + 0.14D_2 + 0.11D_3
\]  \(6.10\)

From now on we refer to Eqns. (6.7) through (6.10) as the AASHO formula. In general the AASHO formula implies that the condition of a pavement deteriorates with accumulated traffic \( W \) and the axle load \( L \). Furthermore the pavement serviceability improves by increasing the thickness index \( D \) that is a weighted sum of layer thicknesses \( D_1', D_2', \) and \( D_3' \).

The AASHO Road Test has been the basis for pavement design practices during the last thirty years. In 1986 AASHTO published the AASHTO Guide for design of Pavement Structures (AASHTO 1986). This document incorporates the original development of the Road Test data with more recent additions relating to sub-surface drainage, materials, reliability, and others. The AASHTO guide accepts the original AASHO formulations as a starting point and adds new factors for effects neglected in the AASHO model.

### 6.3 Preprocessing the Performance Data of the Road Test

The data from the full factorial experiment for flexible pavements have been preserved. The obtained data base contains 9464 records. Each record consists of 11 blocks which respectively represents: loop number, lane number, surface thickness (in), base thickness (in), subbase thickness (in), section number, index day, weighted accumulated axle load applications (1000 applications), axle load (kips), type of axle, and present serviceability index (to one decimal).

The performance data of the Road Test are for test sections under single-axle traffic and sections under tandem-axle traffic. The AASHO model includes the axle type as one of its variables and one formula is developed for both types of traffic (AASHTO 1962). The axle type is a binary variable and there is no meaning to interpolation between values. Using the data for one type of traffic to build an empirical model for the other type of traffic is not reasonable. Furthermore, the estimated parameters for the empirical model developed
for single-axle traffic may be completely different from the parameters for the tandem-axle traffic and adding some factors to the model may not effectively consider the effect of the axle type. Hence for building MC-HARP approximations we partition the performance data based on the axle type and choose to make separate data-based mathematical models for different types of traffic. For the sake of brevity, in the remaining of this chapter we only discuss the data-based approximation procedure for pavements subjected to single-axle traffic. The procedure is similar for tandem traffic.

After separating performance data for tandem-axle traffic from the given data base, for all 164 test sections we have plotted serviceability trends using the given data base and 5-point serviceability trends recorded in Appendix A of the AASHO report (1962). For all sections, we have checked that the given performance data follow the serviceability trends in the AASHO report and removed data points corresponding to overlaid sections. Data from overlaid sections were not used to build the AASHO formula. We have generated a data set with 4788 data points. We refer to this data set as the measured serviceability data set. Each data point has five input values which are three layer thicknesses \((D_1, D_2, D_3)\), axle load \(L\), and weighted accumulated axle load applications \(W\) and one output value which is the present serviceability index \(P\). We have added to this data set the performance data given in the AASHO report and generated a data set containing 5457 data points in the 5-dimensional input space of \((D_1, D_2, D_3, L, W)\) and real line output space of \(P\). From now on we refer to the PSI values in this data base as the measured PSI values.

We smooth the measured serviceability trend for each pavement section using locally weighted regression scatter plot smoothing (LOWESS) procedure (Chambers 1983). LOWESS employs moving weighted least squares with linear approximation. Assume that there are \(T\) pairs \((W_k, P_k)\) in a measured serviceability trend of a typical section and we want to pass a smooth curve through these \(T\) data points for interval \([W_i, W_u]\). First, we choose \(m\) equally spaced points \(W_i\) in \([W_i, W_u]\). Then for each point \(W_i\), we find an interval \([W_i - \Delta_i, W_i + \Delta_i]\) that contains \(q\) nearest neighbors of \(W_i\) in the set \(\{W_k, k = 1, \ldots, T\}\). The interval radius \(\Delta_i\) is the distance from \(W_i\) to its \(qth\) nearest neighbor along the \(W\) axis. We set \(q\) to be \(0.2T\) for the AASHO data base. Then a line \((a + bW = P)\) is fit to the scatter data points using weighted least squares as follows

\[
\minimize_{a, b} \sum_{k=1}^{T} \Psi_i(W_k)(P_k - a - bW_k)^2
\]

(6.11)

where the weight function \(\Psi_i\) corresponding to interval \([W_i - \Delta_i, W_i + \Delta_i]\) is defined as

\[
\Psi_i(z) = \begin{cases} 
1 - \left| \frac{z - W_i}{\Delta_i} \right|^3 & \text{if } \left| \frac{z - W_i}{\Delta_i} \right| < 1.0 \\
0 & \text{otherwise}
\end{cases}
\]

(6.12)

If \((a^*, b^*)\) is the solution of the minimization problem (6.11), then the smooth serviceability \(\tilde{P}_i\) at \(W_i\) is computed as

\[
\tilde{P}_i = a^* + b^*W_i
\]

(6.13)
For each pavement section, we compute smooth PSI for 43 points equally spaced on the W axis from zero to $1.26 \times 10^6$. We have manually adjusted the smooth serviceability trends to start from point (0.0, 4.2), that they be nonincreasing positive curves, and reasonably follow the tails of measured serviceability trends for the PSI values less than 1.5. Figure 6.2 presents the smooth PSI plotted against the measured PSI for all data points. In Fig. 6.2, the smooth PSI is computed for the 4788 data points in the measured serviceability data set. The mean average and standard deviation for the difference between the smooth PSI and measured PSI are -0.008 and 0.206, respectively. Therefore, the smoothing process filters out high frequency disturbances whose root-mean-square amplitudes are about 0.206 from the measured serviceability trends. Figures 6.3(a) and 6.3(b) show the smooth serviceability trends along with the measured serviceability trends for some of 164 sections in the data base. Some of the replicated sections do not have the same serviceability trends because of uncontrolled effects during the Road Test. We refer to the data set containing smooth serviceability trends as the smooth serviceability data set. This set contains 7052 data points corresponding to serviceability trends for 164 pavement sections. Each smooth serviceability trend is defined by its values for 43 equally distant points.

**Note.** In our study we use scatter plots like Fig. 6.2 to study the performance of different models. We use three indices to measure the scatter of plotted data points: the average of the residuals $\bar{e}$, root-mean-square error $RMS$, and the coefficient of determination $R^2$. Assuming that the scatter plot contains $T$ data points and the ordinate and abscissa of the $i$th data point are $y_i$ and $x_i$, respectively. The scatter measures are defined as

$$\bar{e} = \frac{1}{T} \sum_{i=1}^{T} (y_i - x_i)$$

$$RMS^2 = \frac{1}{T} \sum_{i=1}^{T} (y_i - x_i)^2$$
Fig. 6.3(a) Smooth serviceability trends and trends predicted by AASHO formula
Fig. 6.3(b) Smooth serviceability trends and trends predicted by AASHO formula
\[ R^2 = 1 - \frac{\sum_{i=1}^{T} (y_i - x_i)^2}{\sum_{i=1}^{T} (x_i - \bar{x})^2} \]  

(6.16)

where \( \bar{x} \) is the mean average of the \( x \) values. The smaller are \( \bar{x} \) and \( RMS \) and the closer is \( R^2 \) to one, the closer are \( y \) values to \( x \) values.

### 6.4 Performance of the AASHO Empirical Formula

In this section we want to study the performance of the AASHO formula using the performance data measured during the Road Test. In other words we want to observe how well the AASHO formula represents the data used to build it (i.e., its training set). Fig. 6.4 shows the scatter plot of AASHO predicted PSI versus the measured PSI and the smooth PSI with and without considering the replicate section pairs. It is evident that points in the scatter plots do not appear as a diagonal band and funnel in. This observation indicates that the AASHO formula does not represent its training data with a high level of acceptability. The scatter measures for the plots in Fig. 6.4 are shown in the first row of Table 6.2. The coefficient of determination is negative for the scatter plot of the AASHO predicted PSI versus the measured PSI. This observation indicates that the variance of residuals between predicted PSI values and the measured ones is larger than the variance of measured PSI values. With respect to the smooth PSI values, the AASHO formula has \( R^2 \) equal to 0.788 considering the serviceability data of all 164 sections and has \( R^2 \) equals to 0.784 considering the serviceability data of 124 nonreplicate sections. The low \( R^2 \) values indicate the low goodness-of-fit for the AASHO formula. In Fig. 6.3, we have plotted the serviceability trends predicted by the AASHO formula for a number of pavement sections of the Road Test. The AASHO formula does not acceptably represent the measured serviceability trends of a significant number of pavement sections.

For each replicate section pair an adequately fit empirical formula should predict a serviceability trend that passes between the replicate serviceability trends and therefore the smooth serviceability data will deviate from any fit empirical equation. In other words, the root-mean-square error for the plot based on the nonrep-
Table 6.2 Performance of constructed approximations for the training set

<table>
<thead>
<tr>
<th>Approximation</th>
<th>Subdomain approximation</th>
<th>$\varepsilon$</th>
<th>RMS</th>
<th>$R^2$</th>
<th>$\overline{\varepsilon}$</th>
<th>RMS</th>
<th>$R^2$</th>
<th>$\varepsilon$</th>
<th>RMS</th>
<th>$R^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AASHO</td>
<td>---</td>
<td>-0.086</td>
<td>0.671</td>
<td>-0.134</td>
<td>0.136</td>
<td>0.763</td>
<td>0.784</td>
<td>0.129</td>
<td>0.756</td>
<td>0.788</td>
</tr>
<tr>
<td>MC-HARP</td>
<td>Linear</td>
<td>0.10</td>
<td>-0.023</td>
<td>0.279</td>
<td>0.803</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.183</td>
<td>0.988</td>
</tr>
<tr>
<td>MC-HARP</td>
<td>Linear</td>
<td>1.75</td>
<td>-0.185</td>
<td>0.458</td>
<td>0.470</td>
<td>0.027</td>
<td>0.338</td>
<td>0.022</td>
<td>0.388</td>
<td>0.944</td>
</tr>
<tr>
<td>MC-HARP</td>
<td>AASHO</td>
<td>1.75</td>
<td>-0.053</td>
<td>0.385</td>
<td>0.626</td>
<td>-0.007</td>
<td>0.192</td>
<td>0.010</td>
<td>0.260</td>
<td>0.975</td>
</tr>
</tbody>
</table>

plicate sections should be smaller or at most equal to the $RMS$ for the plot considering the replicate section pairs. It is evident that the AASHO formula does not show this characteristic and its $RMS$ value increases by omitting replicate section pairs. This observation indicates that the AASHO predicted serviceability trends for some replicate section pairs do not pass between their pairs of smooth serviceability trends, as shown in Fig. 6.3 for replicate sections 741 and 709.

A review of measured serviceability trends, such as those shown in Fig. 6.3, reveals that their shapes are not what one would have expected for the AASHO formula. Instead of being smooth decreasing curves, the measured serviceability trends appeared to decay sharply at either one or two reasonably well defined events. For example the trends for sections 743, 731, 305, and 327 clearly exhibit these critical events. The locations of these two critical events closely correspond with the periods of spring thaw during the two years of the Road Test (Coree and White 1990). The measured serviceability trends are generally piecewise defined; i.e., the PSI decreases at one rate and then at a much greater rate. A large number of sections fails rapidly during the first freeze/thaw event. A number of sections distinctly deteriorate during the first spring thaw period and then fail at the beginning of the next winter if they cracked at the end of first year or fail during the second freeze/thaw event if they survived the first year uncracked. Only pavement sections with thick structural design or with light axle load exhibit no significant loss of serviceability during the two-year period of the Road Test. It appears that time, as measured by the number of freeze/thaw events, is a significant factor missed in the AASHO formula (Coree and White 1990). Furthermore the seasonal weight factor used for the accumulated load axle applications cannot model the effect of freeze/thaw events. The measured serviceability trends with weighted accumulated axle load applications are still piecewise defined and are not as smooth as the AASHO formula represents them.

During the Road Test there were factors like environmental conditions that inflicted damage to the test pavements but could not be measured or controlled. Therefore the actual pavement system tested during the Road Test had more variables than the five variables used in the simplified AASHO model defined in Eqn. (6.2). By using the simplified model we project the actual mapping representing the pavement system from
its high dimensional space to a five-dimensional space. The projection to low dimensional spaces causes sys-
tematic errors in the measurements. The projectional error behaves like noise in the measurements even if
there is no ambient noise in the measurements. Therefore a simplified model should not perfectly follow the
measurements, otherwise, its performance is noise-dominated and has poor generalization. Based on this
characteristic, we can argue that despite the fact that significant factors like the freeze/thaw event are not con-
sidered in the AASHO simplified model, a data-based approximation with a reasonable performance should
follow the main behavior of the measured serviceability trends. In other words a reasonable model for the
pavement performance should predict serviceability trends that are close to the measured trends and follow
the global behavior of the measured trends but the predicted trends should not perfectly mimic the measured
trends. The AASHO predicted serviceability trends follow the global behavior of the measured trends for a
number of test section like 759, 741, 261, 315, and 297 shown in Fig. 6.3. However for a large number of pave-
ment sections the AASHO predicted trends are not close to the measured trends and do not follow their global
behavior, as shown in Fig. 6.3 for sections 727, 771, 729, 265, and 335.

It is evident that the AASHO model does not represent the observed serviceability trends for pavement
sections trafficked at the Road Test. One main reason for this shortcoming is that environmental effects like
the spring thaw event are not considered in the AASHO formula. Another contributing factor is the functional
form of the AASHO parametric model. The piecewise defined behavior of measured serviceability trends indi-
cates that the actual mapping whose features are contained in the performance data of the Road Test is more
complex than the AASHO approximation whose predicted trends are smooth. A more complex approximate
mapping can represent the data more effectively. Furthermore, significant discrepancy between the AASHO
predicted serviceability trends and measured trends for a number of pavement sections exposes the poor per-
formance of the AASHO formula for some regions of the input domain. A local approximation can model the
pavement performance data for the entire input domain better than a global approximation like the AASHO
formula. An MC-HARP approximation is a nonparametric local mapping and can have more complexity and
adaptivity than the AASHO parametric model. In the following sections we construct MC-HARP approxima-
tions using the performance data of the Road Test and compare their performances with the AASHO formula.
We show that MC-HARP can build a better model for the pavement performance.

6.5 Training and Test Sets

We use all performance data measured during the AASHO Road Test to build MC-HARP approximations.
We associate with the pavement design of a pair of replicate sections the serviceability trends that is the aver-
age of two smooth serviceability trends for each individual section of the pair. The training set contains
smooth serviceability trends for all nonreplicate sections and averages of smooth serviceability trends for
replicate section pairs. By averaging the serviceability trends there remain the serviceability data for 144 un-
duplicated sections whose trends are represented by 43 equally spaced points along the \( W \) axis. The training
set contains present serviceability indices \( P \) for 6192 data points in the input domain of \( (D_1, D_2, D_3, L, W) \).
(From now on a section is defined by a 4-tuple \( (D_1, D_2, D_3, L) \) and we refer to sections belong to training
and test sets as training and test sections, respectively).
We perform a sensitivity-based performance analysis to study the performance of the constructed MC-HARP approximations for points between the training data points. We select a number of points between training data points and use the measured PSI values for their neighboring training data points to evaluate predicted PSI values for test points. The test set contains 6762 points in the input domain corresponding to 322 test sections. The test sections are subjected to the same axle loads used in the Road Test and have layer thicknesses \((D_1, D_2, D_3)\) bounded by layer thicknesses of training sections. Tables 6.3(a) and 6.3(b) shows the specifications of test sections and their corresponding training bounding sections tested during the AASHO Road Test. For example, the test section 1 has the same base thickness, subbase thickness, and axle load as sections 721 and 771 in the AASHO Road Test, but its surface thickness is the average of surface thicknesses for training sections 721 and 771. Sections 721 and 771 are the bounding training sections for the test section 1, as shown in Table 6.3. We do not define any bounding sections for the test sections for which more than one of their layer thicknesses are different from the thicknesses for training sections, such as test sections 34 through 37 as shown in Table 6.3.

The main goal of the sensitivity-based performance analysis is to evaluate the generalization of the constructed approximations. We want to observe how well a data-based mathematical model has captured the main features of the data and the physics behind the Road Test for modeling pavement performance. In the testing phase we look for two trends: (1) the predicted PSI should decrease as the number of accumulated axle load applications increases and (2) the predicted PSI should increase as one of the design variables \((D_1, D_2, D_3)\) increases. From now on we refer to these trends as the testing trends. The first testing trend expresses that the serviceability trends for each test section should be nonincreasing. The second testing trend indicates that the serviceability trend for each test section should be between the serviceability trends of its bounding sections. We recognize a mathematical model like an MC-HARP approximation or the AASHO formula as a reasonable empirical formula for the Road Test, if it adequately represents the training data and follows the testing trends for the majority of test sections.

### 6.6 MC-HARP Approximation for Pavement Performance

In this section we use the proposed MC-HARP method to build a data-based mathematical model for pavement performance using the data of the Road Test. The training set contains \(N = 6192\) data points and the number of test data points is \(N_t = 6192\). The input variables are: the surface, base, and subbase thicknesses \((D_1, D_2, D_3)\), the axle load \(L\), and the logarithm of accumulated single-axle load applications \(\log W\). All input variables are positive real numbers. The maximum values of \(D_1, D_2, D_3, L,\) and \(\log W\) for training and testing data points are 6 in, 9 in, 16 in, 30 kips, and 6.08. The output variable is the present serviceability index \(P\) that is a real number between zero and 5.

We choose the subdomain approximation function \(\theta\) to be a linear function. A term truncation scheme is used to downsize \(\theta\) for small subdomains. The least squares estimator is used to compute the parameters of \(\theta\) during the subdomain training process. A ramp squashing function with \(\mu = 0\) is composed on the subdomain approximation \(\theta\). The MC-HARP sample size \(p\) is set to be 15. For the subdomain partitioning, we use a linear partitioning function with a hierarchy of binary splits. We choose the fuzzy \(\mathcal{P}^\mu\) scheme for selecting
<table>
<thead>
<tr>
<th>Section No.</th>
<th>Thickness (in)</th>
<th>Axle load (kips)</th>
<th>AASHTO Test Bounding Sections</th>
<th>Thickness (in)</th>
<th>Axle load (kips)</th>
<th>AASHTO Test Bounding Sections</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.5</td>
<td>0.0</td>
<td>2.0</td>
<td>721</td>
<td>771</td>
<td>2.0</td>
</tr>
<tr>
<td>2</td>
<td>1.5</td>
<td>0.0</td>
<td>2.0</td>
<td>727</td>
<td>729</td>
<td>2.0</td>
</tr>
<tr>
<td>3</td>
<td>1.5</td>
<td>0.0</td>
<td>2.0</td>
<td>743</td>
<td>759</td>
<td>2.0</td>
</tr>
<tr>
<td>4</td>
<td>1.5</td>
<td>0.0</td>
<td>2.0</td>
<td>717</td>
<td>741</td>
<td>2.0</td>
</tr>
<tr>
<td>5</td>
<td>1.5</td>
<td>0.0</td>
<td>2.0</td>
<td>755</td>
<td>775</td>
<td>2.0</td>
</tr>
<tr>
<td>6</td>
<td>1.5</td>
<td>0.0</td>
<td>2.0</td>
<td>719</td>
<td>737</td>
<td>2.0</td>
</tr>
<tr>
<td>7</td>
<td>2.5</td>
<td>0.0</td>
<td>2.0</td>
<td>771</td>
<td>769</td>
<td>2.0</td>
</tr>
<tr>
<td>8</td>
<td>2.5</td>
<td>0.0</td>
<td>2.0</td>
<td>729</td>
<td>739</td>
<td>2.0</td>
</tr>
<tr>
<td>9</td>
<td>2.5</td>
<td>0.0</td>
<td>2.0</td>
<td>759</td>
<td>773</td>
<td>2.0</td>
</tr>
<tr>
<td>10</td>
<td>2.5</td>
<td>0.0</td>
<td>2.0</td>
<td>741</td>
<td>745</td>
<td>2.0</td>
</tr>
<tr>
<td>11</td>
<td>2.5</td>
<td>0.0</td>
<td>2.0</td>
<td>775</td>
<td>749</td>
<td>2.0</td>
</tr>
<tr>
<td>12</td>
<td>2.5</td>
<td>0.0</td>
<td>2.0</td>
<td>737</td>
<td>763</td>
<td>2.0</td>
</tr>
<tr>
<td>13</td>
<td>1.0</td>
<td>1.0</td>
<td>2.0</td>
<td>721</td>
<td>743</td>
<td>2.0</td>
</tr>
<tr>
<td>14</td>
<td>1.0</td>
<td>1.5</td>
<td>4.0</td>
<td>727</td>
<td>717</td>
<td>2.0</td>
</tr>
<tr>
<td>15</td>
<td>1.5</td>
<td>2.0</td>
<td>4.0</td>
<td>743</td>
<td>755</td>
<td>2.0</td>
</tr>
<tr>
<td>16</td>
<td>1.5</td>
<td>4.5</td>
<td>6.0</td>
<td>717</td>
<td>719</td>
<td>2.0</td>
</tr>
<tr>
<td>17</td>
<td>2.0</td>
<td>1.0</td>
<td>4.0</td>
<td>771</td>
<td>759</td>
<td>2.0</td>
</tr>
<tr>
<td>18</td>
<td>2.0</td>
<td>1.5</td>
<td>4.0</td>
<td>729</td>
<td>729</td>
<td>2.0</td>
</tr>
<tr>
<td>19</td>
<td>2.0</td>
<td>2.5</td>
<td>4.0</td>
<td>759</td>
<td>775</td>
<td>2.0</td>
</tr>
<tr>
<td>20</td>
<td>2.0</td>
<td>3.0</td>
<td>4.0</td>
<td>741</td>
<td>741</td>
<td>2.0</td>
</tr>
<tr>
<td>21</td>
<td>3.0</td>
<td>1.0</td>
<td>4.0</td>
<td>769</td>
<td>775</td>
<td>2.0</td>
</tr>
<tr>
<td>22</td>
<td>3.0</td>
<td>1.5</td>
<td>4.0</td>
<td>739</td>
<td>745</td>
<td>2.0</td>
</tr>
<tr>
<td>23</td>
<td>3.0</td>
<td>2.0</td>
<td>4.0</td>
<td>773</td>
<td>749</td>
<td>2.0</td>
</tr>
<tr>
<td>24</td>
<td>3.0</td>
<td>3.0</td>
<td>4.0</td>
<td>745</td>
<td>763</td>
<td>2.0</td>
</tr>
<tr>
<td>25</td>
<td>1.0</td>
<td>2.0</td>
<td>4.0</td>
<td>721</td>
<td>727</td>
<td>2.0</td>
</tr>
<tr>
<td>26</td>
<td>1.0</td>
<td>3.0</td>
<td>4.0</td>
<td>743</td>
<td>717</td>
<td>2.0</td>
</tr>
<tr>
<td>27</td>
<td>1.0</td>
<td>6.0</td>
<td>4.0</td>
<td>755</td>
<td>719</td>
<td>2.0</td>
</tr>
<tr>
<td>28</td>
<td>2.0</td>
<td>2.0</td>
<td>4.0</td>
<td>775</td>
<td>749</td>
<td>2.0</td>
</tr>
<tr>
<td>29</td>
<td>2.0</td>
<td>2.0</td>
<td>4.0</td>
<td>759</td>
<td>741</td>
<td>2.0</td>
</tr>
<tr>
<td>30</td>
<td>2.0</td>
<td>2.0</td>
<td>4.0</td>
<td>773</td>
<td>745</td>
<td>2.0</td>
</tr>
<tr>
<td>31</td>
<td>3.0</td>
<td>2.0</td>
<td>4.0</td>
<td>769</td>
<td>739</td>
<td>2.0</td>
</tr>
<tr>
<td>32</td>
<td>3.0</td>
<td>3.0</td>
<td>4.0</td>
<td>773</td>
<td>745</td>
<td>2.0</td>
</tr>
<tr>
<td>33</td>
<td>3.0</td>
<td>3.0</td>
<td>4.0</td>
<td>749</td>
<td>763</td>
<td>2.0</td>
</tr>
<tr>
<td>34</td>
<td>1.5</td>
<td>1.5</td>
<td>4.0</td>
<td>88</td>
<td>3.5</td>
<td>4.0</td>
</tr>
<tr>
<td>35</td>
<td>1.5</td>
<td>3.0</td>
<td>4.0</td>
<td>89</td>
<td>3.5</td>
<td>4.0</td>
</tr>
<tr>
<td>36</td>
<td>2.5</td>
<td>1.5</td>
<td>4.0</td>
<td>90</td>
<td>3.5</td>
<td>4.0</td>
</tr>
<tr>
<td>37</td>
<td>2.5</td>
<td>2.5</td>
<td>4.0</td>
<td>91</td>
<td>3.5</td>
<td>4.0</td>
</tr>
<tr>
<td>38</td>
<td>1.5</td>
<td>0.0</td>
<td>4.0</td>
<td>722</td>
<td>772</td>
<td>2.0</td>
</tr>
<tr>
<td>39</td>
<td>1.5</td>
<td>0.0</td>
<td>4.0</td>
<td>728</td>
<td>730</td>
<td>2.0</td>
</tr>
<tr>
<td>40</td>
<td>1.5</td>
<td>0.0</td>
<td>4.0</td>
<td>744</td>
<td>760</td>
<td>2.0</td>
</tr>
<tr>
<td>41</td>
<td>1.5</td>
<td>0.0</td>
<td>4.0</td>
<td>718</td>
<td>742</td>
<td>2.0</td>
</tr>
<tr>
<td>42</td>
<td>1.5</td>
<td>0.0</td>
<td>4.0</td>
<td>728</td>
<td>718</td>
<td>2.0</td>
</tr>
<tr>
<td>43</td>
<td>1.5</td>
<td>0.0</td>
<td>4.0</td>
<td>744</td>
<td>756</td>
<td>2.0</td>
</tr>
<tr>
<td>44</td>
<td>1.5</td>
<td>0.0</td>
<td>4.0</td>
<td>718</td>
<td>720</td>
<td>2.0</td>
</tr>
<tr>
<td>45</td>
<td>2.0</td>
<td>1.0</td>
<td>4.0</td>
<td>772</td>
<td>760</td>
<td>2.0</td>
</tr>
</tbody>
</table>

Table 6.3(a) Test sections and their bounding training sections
<table>
<thead>
<tr>
<th>Thickness (in)</th>
<th>Axle load (kips)</th>
<th>AASHTO Test Bounding Sections</th>
<th>Thickness (in)</th>
<th>Axle load (kips)</th>
<th>AASHTO Test Bounding Sections</th>
<th>Thickness (in)</th>
<th>Axle load (kips)</th>
<th>AASHTO Test Bounding Sections</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0</td>
<td></td>
<td></td>
<td>18.0</td>
<td></td>
<td></td>
<td>30.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12.0</td>
<td></td>
<td></td>
<td>18.0</td>
<td></td>
<td></td>
<td>30.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14.0</td>
<td></td>
<td></td>
<td>18.0</td>
<td></td>
<td></td>
<td>30.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16.0</td>
<td></td>
<td></td>
<td>18.0</td>
<td></td>
<td></td>
<td>30.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18.0</td>
<td></td>
<td></td>
<td>18.0</td>
<td></td>
<td></td>
<td>30.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20.0</td>
<td></td>
<td></td>
<td>18.0</td>
<td></td>
<td></td>
<td>30.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22.0</td>
<td></td>
<td></td>
<td>18.0</td>
<td></td>
<td></td>
<td>30.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24.0</td>
<td></td>
<td></td>
<td>18.0</td>
<td></td>
<td></td>
<td>30.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.3(b) Test sections and their bounding training sections
the splitting thresholds. The maximum training residual is chosen to be the termination criterion for the sub-domain training process. Since the measured PSI values are accurate to one decimal point, the tolerance value $\varepsilon$ for the termination criterion is set to be 0.1.

The constructed MC-HARP approximation with $\varepsilon=0.1$ has 3029 parameters estimated using 6192 training data points. The constructed approximation is more complex than the AASHO formula which has only 11 parameters. However, its number of parameters is well below the number of data. The number of subdomains generated by the HARP partitioning is 843 in average for 15 HARP partitions. The fact that the number of subdomains is large indicates that the data has locally complex behavior for entire input domain. Furthermore, a large number of linear approximations associated with small subdomains are needed to represent the data in different regions of the input domain.

The scatter plots of predicted PSI values against measured and smooth PSI values are shown in Fig. 6.5. The second row of Table 6.2 shows the scatter measures for these plots. The small average and root-mean-square errors and $R^2$ values close to one exhibit perfect fit of training data. The maximum PSI for training and also test data points is 4.2, as shown in Figs. 6.5 and 6.13, and is associated with untrafficked sections; i.e., $W=0$. This observation indicates that the constructed MC-HARP approximation has captured the feature that the PSI value for an untrafficked section is 4.2. The predicted PSI values are close to smooth PSI values without considering replicate section pairs. The predicted serviceability trends follow the average trends for replicate section pairs, as shown in Fig. 6.6. The root-mean-square error of 0.183 for scatter plot of the predicted PSI versus the smooth PSI considering replicate section pairs represents the mean average of deviation between two measured trends of a replicate section pairs. Therefore, the precision tolerance of a smooth PSI value is at least 0.183.

Figure 6.6 shows serviceability trends predicted by the constructed MC-HARP approximation for some of the training sections. The MC-HARP predicted trends perfectly follow the smooth measured trends. The performance of the constructed MC-HARP approximation for training data is superior to the AASHO formula. The MC-HARP approximation computes a deviation measure $\sigma$ for each predicted PSI. The $\sigma$ measure is the standard deviation of the sample of outputs predicted by a sample of $p=15$ HARP approximations for a point

![Fig. 6.5 Performance of the MC-HARP approximation with linear subdomain approximations for the training set $\varepsilon=0.1$.]
Fig. 6.6 Performance of the MC-HARP approximation with linear subdomain approximations for the training set $\varepsilon = 0.1$
in the input domain. A one-standard-deviation confidence interval for a predicted PSI value $P$ is $[p - \sigma, p + \sigma]$. In Fig. 6.6 the narrow shaded strip along each predicted serviceability trend represents its confidence strip for one standard deviation. The average of deviation measures for training data points $\sigma_{\text{Train}}^2$ is 0.012. Therefore the average width of confidence strip is 0.024 which is negligible. The smallness of $\sigma_{\text{Train}}^2$ indicates that the constructed MC-HARP approximation fits the training data with high precision.

Figure 6.7 presents the predicted serviceability trends and their confidence strips for a number of test sections. In general, the predicted trends are nonincreasing and reasonably pass between their bounding trends. The average deviation measure $\sigma_{\text{Test}}^2$ for the test data points 0.779. Therefore, the average width of a one-standard-deviation confidence strip is 1.558. To check the second testing trend that each predicted serviceability trend should be between its bounding trends, one should always consider the confidence strip instead of its mean average, because every possible curve inside a confidence strip has a good probability of being the predicted serviceability trend. One can observe that, for a few test sections, the bounding serviceability trends are inconsistent and cross each other instead of being totally separate, as shown in Figs. 6.7 and 6.12 for test sections 11, 150, 160, and 272. This observation shows the effect of uncontrolled factors on the precision of performance data measured during the Road Test. The AASHO predicted serviceability trends for test sections are plotted in Fig. 6.7. In general, both the MC-HARP approximation and the AASHO formula satisfy the second testing trend for the same number of test sections but not the same sections. The serviceability trends predicted by the AASHO formula are smooth decreasing curves. This behavior is an inherent characteristic of the AASHO formula. The AASHO formula satisfies the first testing trend better than the constructed MC-HARP approximation. The constructed MC-HARP approximation with $\varepsilon = 0.1$ has superior performance to the AASHO empirical model for the training data but does not satisfies testing trends sought for test sections as acceptable as the AASHO formula.

6.6.1 Smoothing Predicted Serviceability Trends

Preparing more data is the best way to improve the performance of a data-based empirical model like an MC-HARP approximation. Performance data can implicitly force the approximation to satisfy testing trends. Unfortunately no more data are available for the AASHO problem. However we can explicitly smooth the undesirable local disturbances in the MC-HARP predicted serviceability trends. We use a linear spline functions with Gauss-Markov least-squares estimator as the smoother. Assume for a pavement section, the constructed MC-HARP approximation has predicted PSI values $\{P_i\}_{i=1}^T$ for $T$ accumulated axle load applications $\{W_i\}_{i=1}^T$. A linear spline $B(W; v)$ with $m$ knots is passed through the predicted serviceability trend by solving the following constrained optimization problem

$$
\begin{align*}
\text{minimize} & \quad \sum_{i=1}^{T} \frac{1}{\sigma(W_i)} [B(W_i; v) - P_i]^2 \\
\text{subject to} & \quad v_i \geq v_{i+1} \quad i = 1, \ldots, m \\
& \quad B(0; v) = 4.2
\end{align*}
$$

(6.17)
Fig. 6.7 Performance of the MC-HARP approximation with linear subdomain approximations for the test set $e=0.1$
where \( v_i \) is the \( i \)th degree of freedom associated with the \( i \)th knot located at \( W_i \) and \( \sigma(W_i) \) is the deviation measure predicted by the MC-HARP approximation for the \( i \)th PSI value. The filtered PSI value for \( W_i \) is \( B(W_i; v^*) \) where \( v^* \) is the solution for the optimization problem. In the smoothing problem (6.17), the inequality constraints restrict the optimum spline to be nonincreasing and the equality constraints make the filtered PSI equal to 4.2 for an untrafficked pavement (like the AASHO formula). The objective function is a fitness index computed as the weighted sum of square residuals between predicted PSI values and their corresponding filtered PSI values. By minimizing the objective function, we want to find the closest spline to the predicted serviceability trend. The weight values in the proposed fitness index force the optimal spline to be closer to the points with higher confidence indices (lower deviation measures) on the predicted trend.

We solve problem (6.17) using an optimization algorithm developed for nonlinear parameter estimation of structural systems (Banan 1993). The algorithm is based on recursive quadratic programming (RQP) and can solve constrained nonlinear optimization problems. The RQP algorithm is globally convergent and is amenable to large-scale computation. The selected linear spline has 12 knots at points 0, 60 \( \times 10^3 \), 120 \( \times 10^3 \), 180 \( \times 10^3 \), 240 \( \times 10^3 \), 300 \( \times 10^3 \), 450 \( \times 10^3 \), 600 \( \times 10^3 \), 750 \( \times 10^3 \), 1050 \( \times 10^3 \), 1200 \( \times 10^3 \) along the \( W \) axis. We have filtered all 322 predicted serviceability trends for test sections. Fig. 6.8 shows filtered trends for test sections whose original predicted trends are shown in Fig. 6.7. The filtered predicted serviceability trends are smoother, nonincreasing, and follow the region between the bounding trends better than the unfiltered predicted serviceability trends.

### 6.7 Building an MC-HARP Approximation with Suitable Complexity

As we mentioned in Section 6.4, an empirical model that perfectly fits the training serviceability trends is over-parameterized and has poor generalization. Although we have shown that spline smoothing can improve the performance of such a model, we can significantly improve performance by selecting suitable complexity for the constructed model. In the following we use the MC-HARP model selection technique and framework for classifying data sets to build an MC-HARP approximation with suitable complexity for pavement performance.

Figure 6.9 shows the performance curves for the MC-HARP approximation with linear subdomain approximations. In Fig. 6.9, we have also shown the root-predicted-error \( RPS_F \) curve by setting the standard deviation of noise to be 0.6. The \( \sigma_{Train}^F \) curve is skewed to the right. This observation indicates that the amplitude of noise is small; i.e., \( \lambda < \lambda^* \). The optimal tolerance \( \varepsilon^* \) cannot be located. No local minimum is developed on both \( \sigma_{Train}^F \) and \( \sigma_{Test}^F \) curves. Hence the amount of data is not adequate; i.e., \( N < N^* \). Consequently the conditioning of the data set and the confidence in approximation are not measurable. However by playing with the amount of training data, we have observed that the location and values of maximum points of curves \( RMS_{Train}^F \) and \( \sigma_{Train}^F \) do not significantly vary by changing the amount of data. This observation leads to the conclusion that the amount of data is greater than \( N_{min} \). Since the amplitude of noise \( \lambda \) is less than \( \lambda^* \) and \( N_{min} < N < N^* \), by following the decision tree of the MC-HARP framework for classifying data sets, it can be concluded that the Road Test data set belongs to the \( QQ_1 \) quality-quantity region.
Fig. 6.8 Performance of the filtered MC-HARP approximation with linear subdomain approximations for the test set $e = 0.1$
The critical tolerance values $\varepsilon_0$, $\varepsilon_1$, $\varepsilon_2$, and $\varepsilon_3$ are 1.75, 1.75, 2.2, and 2.7, respectively. The bounds for the optimal tolerance value $\varepsilon^*$ and the selected value for the most suitable tolerance value $\varepsilon_s$ are determined by using the MC-HARP rules for selecting suitable complexity for data sets in the quality-quantity region $QQ_1$. The optimal tolerance belongs to the interval $[1.75, 2.2]$. A reliable selected value for $\varepsilon_s$ cannot be determined but by setting $\varepsilon_s$ to be equal to $\varepsilon_0 = 1.75$ we prevent building an approximation with unreasonable complexity. We adjust the estimate for the noise amplitude in the $RPS_F$ measure such that the local minimum of the $RPS_F$ curve is located at the selected tolerance value, as shown in Fig. 6.9. The estimated amplitude of noise is equal to 0.6, a value much smaller than the average of measured present serviceability indices for the training data points. This observation supports the conclusion that the amount of noise is small.

The constructed MC-HARP approximation with $\varepsilon = 1.75$, $\bar{F}(1.75)$, has 693 parameters and 116 subdomains on average for 15 HARP partitions. The constructed approximation is much simpler than the MC-HARP approximation with $\varepsilon = 0.1$, $\bar{F}(0.1)$. The scatter plots of predicted PSI values against measured and smooth PSI values are shown in Fig. 6.10. It is evident that points in the scatter plots are close to the diagonal and
there is a curvature in the scatter plots. (The curvature can be eliminated by composing a nonlinear univariate function on the output of the constructed approximation. The nonlinear output function could improve the performance of the constructed MC-HARP approximation but we did not use it for the performance analysis).

The maximum PSI for the training and also test data points is about 4.2, as shown in Figs. 6.10 and 6.13, and is associated with untrafficked sections; i.e., \( W = 0 \). This observation indicates that \( \overline{P}(1.75) \) has captured the feature that the PSI value for an untrafficked section is 4.2. The third row of Table 6.2 shows the scatter measures for scatter plots. The small average and root-mean-square errors and \( R^2 \) values close to one exhibit good fit of training data. For training data, the MC-HARP approximation with \( \varepsilon = 1.75 \) does not fit the data as perfectly as \( \overline{P}(0.1) \) but its performance is superior to the AASHO formula.

Figure 6.11 and 6.12 show serviceability trends and their confidence strips predicted by the constructed MC-HARP approximation with \( \varepsilon = 1.75 \) for a number of training and test sections. The MC-HARP predicted trends closely but did not perfectly follow the smooth measured trends. The deviation measure for training data points \( \sigma_p^{\text{Train}} \) is 0.349, equal to the root-mean-square error for the training set. The average width of the confidence strip is 0.698. The constructed MC-HARP empirical model represents the training data with good precision and accuracy. For test sections, the predicted serviceability trends are almost nonincreasing and pass between their bounding trends for the majority of test sections. The deviation measure for the test set \( \sigma_p^{\text{Test}} \) is 0.639 and consequently the average width of confidence strips is 1.278. In comparison to the \( \overline{P}(0.1) \) approximation, the \( \overline{P}(1.75) \) approximation has better approximation precision for test data points and its predicted trends are smoother and more likely to be nonincreasing and they satisfy the second testing trend better. Furthermore the serviceability trends predicted by the MC-HARP approximation with \( \varepsilon = 1.75 \) pass between their bounding trends for more test sections than the AASHO formula.

By selecting a suitable complexity, we simplify the constructed MC-HARP approximation and improve its performance for test data points without significantly degrading its performance for the training set. In Fig. 6.13 we compare the performance of constructed MC-HARP approximations \( \overline{P}(1.75) \) and \( \overline{P}(0.1) \). Since \( \overline{P}(0.1) \) perfectly fits the training data, the scatter plot 6.13(a) is similar to the scatter plots in Fig. 6.10 for smooth PSI values. In other words the PSI values predicted by \( \overline{P}(0.1) \) represent the training set and Fig. 6.13(a) is actually showing the measured smooth PSI versus the PSI predicted by \( \overline{P}(1.75) \). The MC-HARP approxima-
Fig. 6.11(a) Performance of the MC-HARP approximation with linear subdomain approximations for the training set $\varepsilon = 1.75$
Fig. 6.11(b) Performance of the MC-HARP approximation with linear subdomain approximations for the training set $\varepsilon = 1.75$
Fig. 6.11(c) Performance of the MC-HARP approximation with linear subdomain approximations for the training set $e=1.75$
Fig. 6.12(a) Performance of the MC-HARP approximation with linear subdomain approximations for the test set $\epsilon = 1.75$
Fig. 6.12(b) Performance of the MC-HARP approximation with linear subdomain approximations for the test set $\varepsilon = 1.75$
Fig. 6.12(c) Performance of the MC-HARP approximation with linear subdomain approximations for the test set $\varepsilon = 1.75$
tion $F(1.75)$ is closer to the MC-HARP approximation $F(0.1)$ for the test set than for the training set. This observation can be supported by the fact that the scatter plot 6.13(b) has lower RMS error and higher $R^2$ measure than the scatter plot 6.13(a). The scatter shown in Fig. 6.13(b) represent the difference between $F(1.75)$ and $F(0.1)$ for the test set and the performance distance of $F(0.1)$ from the better performance exhibited by $F(1.75)$. By reducing complexity we smooth $F(0.1)$, filter out some of its local disturbances, and build the MC-HARP approximation $F(1.75)$ that has better performance.

6.8 MC-HARP Approximation with the AASHO Subdomain Approximation

The experimenter may choose one possible family of functions for the subdomain approximation $\theta$ when theory, past experience and/or other sources are available that provide detailed knowledge about the form of the actual mapping whose features are contained in the data. A priori knowledge of the physics of the mathematical modeling problem can be used to restrict the structure of $\theta$. Conversely by selecting specific functional form for the parametric approximation $\theta$, a priori knowledge about the actual model can be implemented in the constructed MC-HARP approximation.

We can improve the performance of an MC-HARP approximation by using a priori knowledge about the physics of the pavement performance modeling problem. For the task of building an empirical model for the long term pavement performance, we know two trends about the performance of a reasonable model. These trends, called testing trends, are: (1) the predicted serviceability trend for a pavement should be a nonincreasing curve and (2) the predicted PSI should increase by increasing the overall thickness of a pavement structure and by decreasing of the axle load. By using a parametric subdomain approximation $\theta$ that satisfies the testing trends, we build an MC-HARP approximation that inherently follows the testing trends and does not need to learn them from the data. A good candidate for such a subdomain approximation is the AASHO parametric model. The AASHO formula intrinsically satisfies the testing trends but cannot represent the performance data of the Road test in a best way. The MC-HARP method can be used to increase the adaptivity of the AASHO model. By using the AASHO parametric form for the subdomain approximation of an MC-HARP approximation, a mutual improvement is expected. MC-HARP helps the AASHO formula to represent data in a better way and on the other hand, the AASHO model helps an MC-HARP approximation to reliably capture the testing trends, improve its generalization, and does not show unrealistic performance outside the training set.

We choose the subdomain approximation function has the AASHO parametric functional form defined in Eqns. (6.3) through (6.6). The selected subdomain approximation takes the form

$$\theta_A(x; w) = 4.2 - 2.7 \left( \frac{x_5}{\varrho (x; w)} \right) \beta (x; w)$$

(6.18)

where the parametric functions $\beta$ and $\varrho$ are

$$\varrho (x; w) = \frac{w_5(D + w_4)^{w_1}}{(x_4 + 1)^{w_7}}$$

(6.19)
where $D = w_1 x_1 + w_2 x_2 + w_3 x_3$, the input variables $x_1$ through $x_5$ are $D_1, D_2, D_3, L$, and $W$, respectively and the parameters $w_1$ through $w_11$ are respectively the coefficients $a_{1-4}, A_{o-2}, B_{o-2}$, and $\beta_o$ used in Eqns. (6.3) through (6.6). We refer to $\theta_A(x; w)$ as the AASHO subdomain approximation. The $\theta_A(x; w)$ approximation has eleven unknown parameters that are estimated during the subdomain training process using the least squares estimator. The subdomain parameter estimation problem for a subdomain containing $T$ data points can be expressed as follows

$$\text{minimize}_{w} \sum_{i=1}^{T} [\theta_A(x_i; w) - P_i]^2$$

subject to $10^{-6} \leq w_j \leq b_j$ \hspace{1cm} $j = 1, \ldots, 11$

where $P_i$ is the measured PSI for the $i$th data point $x_i$ and the upper bounds $\{b_i\}_{i=1}^{11}$ are set to be $\{1, 1, 1, 2, 10^7, 30, 10, 1, 6, 10, 1\}$, respectively. The parameters of $\theta_A$ are constrained to be non-negative. Such a constraint is explained on the basis that $\theta_A$ should satisfy the testing trends. For example the layer coefficients $w_{1-3}$ are positive because an increasing layer thickness must lead to an increase in the overall serviceability. The AASHO subdomain approximation and consequently, the objective function of the estimation problem (6.21) are highly nonlinear with respect to optimization variables $w$. We use recursive quadratic programming to solve the constrained nonlinear minimization problem (6.21). The initial values for the parameters $w$ are set to be the values recommended by the AASHO formula which are $\{0.44, 0.14, 0.11, 1.0, 10^{5.93}, 9.36, 4.79, 0.081, 5.19, 3.23, 0.4\}$, respectively.

No term truncation scheme is developed to downsize $\theta_A$ for small subdomains. Thus to prevent the over-parameterization of $\theta_A$, the subdomain partitioning process are terminated when a subdomain contains less data points than twice the number of parameters in $\theta_A$ which is 22. With this arrangement a binary split does not develop offsprings containing less than 11 data points. A ramp squashing function with $\mu = 0$ is composed on the AASHO subdomain approximation $\theta_A$. The MC-HARP sample size $p$ is set to be 15. For the subdomain partitioning, we use a linear partitioning function with a hierarchy of binary splits. We choose the fuzzy $\mathcal{P}_H$ scheme for selecting the splitting thresholds. The maximum training residual is chosen to be the termination criterion for the subdomain training process.

We use the MC-HARP model selection technique and framework for classifying data sets to build an MC-HARP approximation with suitable complexity. Figure 6.14 shows the performance curves for the MC-HARP approximation with AASHO subdomain approximations, $\bar{F}_A$. Unlike an MC-HARP approximation with linear subdomains, the performance index $RMS_{\text{Train}}^F$ is greater than zero and does not change significantly for tolerance values less than 0.8. Also for the tolerance interval $[0.0, 0.8]$, the deviation measure $\sigma_{\text{Train}}^F$ is almost constant and the deviation measure $\sigma_{\text{Test}}^F$ significantly increases as $\varepsilon$ converges to zero. The reason for these observations is that the training process with no term truncation scheme for $\theta_A$ and early termination from subdomain partitioning process is not locally convergent. Therefore by decreasing the tolerance value $\varepsilon$ to-
ward small values, only large subdomains are partitioned to small offsprings but current small subdomains are not partitioned and their training error does not decrease. Consequently smoothness of the constructed approximation decreases (that leads to increase in the deviation measure $\sigma_{F}^{Test}$) without significant decrease of the training error $RMS_{F}^{Train}$ and deviation measure $\sigma_{F}^{Train}$. Using a locally nonconvergent training process alters the shape of performance curves for only small tolerance values; i.e., the interval $[0, 0.8]$ and does not influence these curves for other tolerance values. The actual shape of performance curves for small tolerance values can be extrapolated from performance curves associated with locally convergent MC-HARP approximations, for example those shown in Fig. 6.9.

In Fig. 6.14 the $\sigma_{F}^{Train}$ curve is skewed to the right. This observation indicates that the amplitude of noise is small; i.e., $\lambda < \lambda^{*}$. Furthermore we can conclude that the AASHO parametric model does not have the same functional form as the actual mapping whose features are contained in the data. The optimal tolerance $\varepsilon^{*}$ cannot be located. No local minimum is developed on both $\sigma_{F}^{Train}$ and $\sigma_{F}^{Test}$ curves. Hence the amount of data is not adequate; i.e., $N < N^{*}$. Consequently the conditioning of the data set and the confidence in approximation are not measurable. By playing with the amount of training data, we have observed that the location and values of maximum points of curves $RMS_{F}^{Train}$ and $\sigma_{F}^{Train}$ do not significantly vary by changing the amount of data. This observation leads to the conclusion that the amount of data is greater than $N_{min}$. Since the amplitude of noise $\lambda$ is less than $\lambda^{*}$ and $N_{min} < N < N^{*}$, it can be concluded that the MC-HARP data-based mapping approximation with the AASHO subdomain approximation belongs to the $QQ_{1}$ quality-quantity region. This result is similar to the classification concluded by the MC-HARP approximation with a linear subdomain approximation.

The critical tolerance values $\varepsilon_{1}$ and $\varepsilon_{3}$ cannot be located on performance curves. The critical tolerances $\varepsilon_{o}$ and $\varepsilon_{2}$ are 1.75 and 3.0, respectively. The bounds for the optimal tolerance value $\varepsilon^{*}$ and the selected value for the most suitable tolerance value $\varepsilon_{s}$ are determined by using the MC-HARP rules for selecting suitable complexity for data sets in the quality-quantity region $QQ_{1}$. The optimal tolerance belongs to the interval $[1.75, 3.0]$. A reliable selected value for $\varepsilon_{s}$ cannot be determined but by setting $\varepsilon_{2}$ to be equal to $\varepsilon_{o}=1.75$ we
prevent building an approximation with unreasonable complexity. We adjust the estimate for the noise amplitude in the $RPS_F$ measure such that the local minimum of the $RPS_F$ curve is located at the selected tolerance value, as shown in Fig. 6.14. The computed estimate for the noise amplitude is equal to 0.4, a value much smaller than the average of measured present serviceability indices for the training data points. This observation supports the conclusion that the amount of noise is small. The estimated amplitude of noise is close to the value 0.6 estimated using the MC-HARP approximation with linear subdomains.

The MC-HARP approximation with the AASHO subdomain approximation for $\varepsilon = 1.75$, $F_A(1.75)$, has 602 parameters and 55 subdomains in average for 15 HARP partitions. Figure 6.15 shows complexity curves for MC-HARP approximations. The MC-HARP approximation $F_A(1.75)$ is slightly simpler and has fewer subdomains than the MC-HARP approximation with linear subdomains $F(1.75)$. The number of parameters for $F_A(1.75)$ significantly increases for tolerance values less than 1.2 partially because the training process is not locally convergent.

The scatter plots of predicted PSI values against measured and smooth PSI values are shown in Fig. 6.16. Points in the scatter plots are clustered along the diagonal. This observation suggests a good fit of the training data. The maximum PSI for the training and also test data points is 4.2, as shown in Figs. 6.16 and 6.19, and

![Complexity curves for MC-HARP approximations](image)

**Fig. 6.15** Complexity curves for MC-HARP approximations

![Performance of the MC-HARP approximation with the AASHO subdomain approximation for the training set $\varepsilon = 1.75$](image)

**Fig. 6.16** Performance of the MC-HARP approximation with the AASHO subdomain approximation for the training set $\varepsilon = 1.75$
is associated with untrafficked sections; i.e., \( W=0 \). This property is inherent in the AASHO subdomain approximation and does not need to be learned from data. The fourth row of Table 6.2 shows the scatter measures for scatter plots. The small average and root-mean-square errors and \( R^2 \) values close to one exhibit good fit of training data. According to Table 6.2, for training data, the MC-HARP approximation with the AASHO subdomain approximation fits the data better than the MC-HARP approximation \( F(1.75) \) and its performance is superior to the AASHO formula.

Figure 6.17 and 6.18 show serviceability trends and their confidence strips predicted by the constructed MC-HARP approximation with the AASHO subdomain approximation for a number of training and test sections. The MC-HARP predicted trends closely follow the smooth measured trends. The deviation measure for training data points \( \sigma^\text{Train}_F \) is 0.106. The average width of confidence strip is 0.212. Hence the constructed MC-HARP empirical model represents the training data with good precision and accuracy. Furthermore the approximation \( F_A(1.75) \) has better approximation accuracy (lower \( \text{RMS}^\text{Train}_F \) and precision (lower \( \sigma^\text{Train}_F \)) for the training set than the MC-HARP approximation with linear subdomain approximations \( F(1.75) \). For test sections, the predicted serviceability trends are almost nonincreasing and pass between their bounding trends for the majority of test sections with good confidence. The deviation measure for the test set \( \sigma^\text{Test}_F \) is 0.430 and consequently the average width of confidence strips is 0.860. In comparison to the \( F(1.75) \) approximation, the \( F_A(1.75) \) approximation has better approximation precision (lower \( \sigma^\text{Test}_F \) for test data points and its predicted trends are smoother and more nonincreasing and they satisfy the second testing trend better. Furthermore the serviceability trends predicted by the MC-HARP approximation with the AASHO subdomain approximation pass between their bounding trends for more test sections than the AASHO formula.

By selecting a suitable subdomain approximation we improve the performance of the constructed MC-HARP approximation for test and training data points. In Fig. 6.19 we compare the performance of constructed MC-HARP approximations \( F_A(1.75) \) and \( F(1.75) \). The scatter plot 6.19(a) is similar to the scatter plots in Fig. 6.10 for smooth PSI values and has the curvature associated with \( F(1.75) \). This similarity indicates that the PSI values predicted by \( F_A(1.75) \) represent the smooth measured PSI values. The same similarity can be observed for the scatter plot 6.19(b). These similarities suggest that the performance of \( F_A(1.75) \) for
Fig. 6.17(a) Performance of the MC-HARP approximation with the AASHO subdomain approximation for the training set $\varepsilon = 1.75$
Fig. 6.17(b) Performance of the MC-HARP approximation with the AASHO subdomain approximation for the training set $\varepsilon = 1.75$
Fig. 6.17(c) Performance of the MC-HARP approximation with the AASHO subdomain approximation for the training set $\varepsilon = 1.75$. 

- AASHO Formula
- MC-HARP Approximation
- Measured PSI
- Confidence Strip
- Replicate Sections
Fig. 6.18(a) Performance of the MC-HARP approximation with the AASHO subdomain approximation for the test set $\varepsilon = 1.75$. 

- AASHO Formula
- Confidence Strip
- MC-HARP Predicted Trend
- Bounding Trend

Weighted Axle Load Applications, 1000's
Fig. 6.18(b) Performance of the MC-HARP approximation with the AASHO subdomain approximation for the test set $\varepsilon = 1.75$
Fig. 6.18(c) Performance of the MC-HARP approximation with the AASHO subdomain approximation for the test set \( \varepsilon = 1.75 \)
the test set is close to its performance for the training data. Furthermore the MC-HARP approximation with the AASHO subdomain approximation has captures the main features of the data and generalizes them to other points in the input domain. The scatter plot 6.19(a) has lower RMS error and higher $R^2$ measure than the scatter plot 6.19(b). This observation indicates that the MC-HARP approximation $\overline{F}(1.75)$ is closer to the MC-HARP approximation $\overline{F}_A(1.75)$ for the training set than for the test set. The performance of $\overline{F}(1.75)$ is not far from the performance of $\overline{F}_A(1.75)$. Therefore the MC-HARP method is able to build an approximation, $\overline{F}(1.75)$, that captures the main features of the data and has good generalization without using a priori knowledge. The scatter shows in Fig. 6.19(b) represent the difference between $\overline{F}_A(1.75)$ and $\overline{F}(1.75)$ for the test set and the performance distance of $\overline{F}(1.75)$ from the better performance exhibited by $\overline{F}_A(1.75)$. By selecting a subdomain approximation that inherently exhibits some global feature of the data, we build the MC-HARP approximation $\overline{F}_A(1.75)$ that is better than the MC-HARP approximation that not using a priori knowledge and is superior to the AASHO empirical model.

### 6.9 Verification of the AASHO Formula Using MC-HARP

The AASHO formula is a global parametric model whose parameters are determined using the performance data of the Road Test. The AASHO model has a valid structure selected based on the physics of pavement performance. Furthermore it can reasonably mimic the performance of a pavement system tested during the Road Test. Therefore the AASHO model is a valid and reasonably approximate model for pavement performance. However like any global approximation, it cannot represent the nonhomogeneous data of the Road test in a best way due to its low local adaptivity. We have shown that MC-HARP can increase the adaptivity of the AASHO model and improve its performance. The MC-HARP method associates the AASHO model with subdomains of the input domain instead of the entire input domain. The suitability of the AASHO formula for different regions of the input domain can be verify by observing how its parameters vary for different subdomains.

The constructed MC-HARP approximation is the average of a sample of 15 HARP approximations. Each HARP approximate mapping is a local approximation that partitions the input domain into 55 subdomains, on average. The constructed approximation for each subdomain has the functional form of the AASHO model and its parameters are estimated using the data points in the subdomain. Therefore different subdomains may have different parameters. Each parameter has about 55 estimated values for each HARP approximation. We use these values to compute the histogram of each estimated parameter. Then we average 15 histograms associated with 15 HARP approximation. Figure 6.20 shows the mean average histograms for all parameters of the AASHO formula. The domain of the histogram corresponding to the $i$th parameter $w_i$ is the bounding interval for $w_i$ defined in Eqn. (6.21). The number of bins for each histogram is set to be 80. Each histogram is normalized such that the area under it is equal to the upper bound of its domain.

All histograms in Fig. 6.20 have multiple peaks except for the parameter $A_1$. The peaks developed at the bounding values of a parameter represent the sum of frequencies for values greater than the upper bound or smaller than the lower bound. By increasing upper bounds or decreasing lower bounds (if it is plausible), the end peaks may flatten. The selected bounding intervals for each parameter is plausible and contains the AA-
Fig. 6.20 Distribution of parameters in the AASHO formula

* AASHO recommended value
SHO values. If the AASHO model is suitable for all subdomains of the input domain then all histograms should have only one peak located at the value recommended by the AASHO formula. This characteristic is not observed in Fig. 6.20. There are multiple peaks for each histogram. All histograms have one peak at the value recommended by the AASHO formula. This observation indicates that for a significant number of subdomains in the input domain the AASHO model is the best approximation. Since the maximum frequency associated with the AASHO formula is not the global maximum for parameters $a_{1-3}, A_0, A_2, B_0, B_1,$ and $\beta_0,$ the AASHO formula is not the most suitable approximation for the majority of subdomains. The peaks of histograms are sharp and distinct. Therefore the estimated parameters have only a few best values for all subdomains. The most likely values for parameters $a_{1-4}, A_{0-2}, B_{0-2},$ and $\beta_0$ are (1, 0.5, 0.5, 1, 0.5x10^7, 9.36, 10, 0.5, 6.0, 3.23, 0, ), respectively. These values are different from the AASHO recommended values except for parameters $a_4, A_1,$ and $B_2.$ A parametric approximation with the most likely parameter values may not be the best approximation for the majority of subdomains. In other words the maximum likelihood estimate of the parameters vector may not be the same as the vector of most likely parameters because the parameters are not independent from one another.

The distribution of estimated parameters of the AASHO subdomain approximations developed by MC-HARP have been used to verify that to what extent the AASHO model is suitable for representing the performance data. The parameter values selected for the AASHO formula are not suitable for the entire input domain. These parameters need to be varied for different regions of the input domain in order to better represent the data of the Road Test. The MC-HARP method has shown that by partitioning the input domain into a small number of subdomains, the performance of the AASHO model significantly improves.

6.10 Conclusions

In this chapter we have shown that as a result of the AASHO Road Test, a data-based mathematical model can be built for the long term performance of a pavement. Such a model can be used for the design and performance analysis of pavement structures.

We have used a sensitivity-based performance analysis to observe how well a data-based mathematical model has captured the main features of the data and the physics behind the Road Test for modeling pavement performance. We recognize a mathematical model to be a reasonable empirical formula for the Road Test if it adequately represents the training data and follows two testing trends for the majority of test sections. The testing trends are: (1) the predicted serviceability trend for a pavement should be a nonincreasing curve and (2) the predicted PSI should increase by increasing the overall thickness of a pavement structure and by decreasing of the axle load.

We have studied the performance of the model that is developed in the AASHO report (1962) and has been the basis for pavement design practice during the last thirty years. The AASHO model does not represent the observed serviceability trends for pavement sections trafficked at the Road Test in the best way. Significant discrepancy between the AASHO predicted serviceability trends and measured trends for a number of pavement sections exposes the poor performance of the AASHO formula for some regions of its input domain.
It has been concluded that a local approximation like an MC-HARP approximation can model the pavement performance data for the entire input domain better than a global approximation like the AASHO formula.

We have used the MC-HARP method to build a data-based mathematical model for pavement performance using the data of the Road Test and compared its performance with the AASHO model. An MC-HARP approximation with large complexity perfectly fits the data but its performance for the test set is not as acceptable as the AASHO formula. Spline smoothing can improve the performance of such an approximation. We have used the MC-HARP model selection technique and framework for classifying data sets to build an MC-HARP approximation with suitable complexity for pavement performance. By selecting a suitable complexity, we simplify the constructed MC-HARP approximation and improve its performance for test data points without significantly degrading its performance for the training set. The constructed MC-HARP empirical model represents the training data with good precision and accuracy. Furthermore its performance for the Road Test data is superior to the AASHO formula and satisfies the testing trends for more test sections than the AASHO formula.

We improve the performance of an MC-HARP approximation by selecting a subdomain approximation that inherently exhibits a priori knowledge about the physics of the pavement performance modeling problem. We use the MC-HARP method to construct an MC-HARP approximation whose parametric subdomain approximation is the AASHO model. For training and test sets, the constructed empirical model is better than the MC-HARP approximation with linear subdomain approximations (that are not using a priori knowledge) and is superior to the AASHO empirical model. MC-HARP helps the AASHO formula to better represent data and on the other hand, the AASHO model helps an MC-HARP approximation to reliably capture the testing trends, improve its generalization, and curb unrealistic performance outside the training set.

We have shown that a priori knowledge about the physics of a mathematical modeling problem can be used to restrict the structure of the subdomain approximation of MC-HARP. Furthermore by selecting specific functional form for the parametric subdomain approximation, a priori knowledge about the actual mapping can be implemented in the constructed MC-HARP approximation. The applicability of the MC-HARP method for nonlinear subdomain approximations is shown by using the highly nonlinear AASHO subdomain approximation.

The suitability of the AASHO formula for different regions of the input domain have been verified by observing how its parameters vary for different subdomains developed by MC-HARP. We conclude that the parameters selected for the AASHO formula are not suitable for the entire input domain. Furthermore the values of the parameters in the AASHO formula need to be varied for different regions of the input domain in order to better represent the data of the Road Test.
CHAPTER SEVEN
Closure

"Although this may seem a paradox, all exact science is dominated by the idea of approximation."
Bertrand Russell

Scientific databases can be viewed as critical repositories of knowledge, both existing and yet to be discovered. Current data analysis technology falls short of supporting the diverse need of scientific and engineering applications. We have overcome these shortcomings with the MC-HARP strategy for building data-based information processing system with certain brain-like functionalities. A novel combination of parallel distributed processing, fuzzy logic, and the Monte Carlo strategy was used to develop the MC-HARP environment. The developed data-based information processing system is self-organizing and hence adaptive and has the ability to generalize from its data. It can process heterogeneous data and operate while requiring minimal external adjustment. It can interactively accept knowledge and provide guidance for efficiently improving the database. MC-HARP capitalizes on distributed parallel computing to have computationally efficient self-organization and execution and to be fault tolerant.

We have established the mathematical basis for building the MC-HARP data-processing environment. The MC-HARP strategy determines the functional structure and parameters of a mathematical model simultaneously. A Monte Carlo (MC) strategy combined with the concept of Hierarchical Adaptive Random Partitioning (HARP) and fuzzy subdomains determines the multivariate parallel distributed mappings. The constructed mapping can be modeled as a neural network. The HARP algorithm is based on a divide-and-conquer strategy that partitions the input space into measurable connected subdomains and builds a local approximation for the mapping task. Fuzziness promotes continuity of the mapping constructed by HARP and smooths the mismatching of the local approximations in the neighboring subdomains. The Monte Carlo superposition of a sample of random partitions, reduces the localized disturbances among the fuzzy subdomains, controls the global smoothness of the mean average mapping, and improves the generalization of the network.

The tree structure of the HARP modules and the independence of both the subdomain approximations and the random partitions enable the MC-HARP environment to quickly converge to a series of equally plausible solutions without user interaction. The MC-HARP environment enjoys a large-scale granularity produced by the Monte Carlo parallelism and the geometric parallelism achieved by partitioning the input space. Therefore this environment can exhibit good performance on parallel computers for large and complex scientific databases.

The developed MC-HARP philosophy for building data-based approximate mappings leads to a novel model selection criterion and an original framework for classifying data-fitting problems. The MC-HARP en-
vironment not only can build approximate multivariate mappings with self-organization capability, noise and fault tolerance, adaptivity, generalization, highly plastic and stable learning characteristics with respect to the addition of new data points, and parallel structure but also can answer fundamental questions in data-based mathematical modeling. These questions include:

- What is the confidence level for each predicted output of the constructed model?
- What is the approximation confidence measure for the constructed model?
- How does the functional complexity of the actual multivariate mapping change over the input space?
- What is the suitable structural complexity for a data-based model using noisy data?
- What is the level of noise in the data?
- Is the amount of training data adequate? If not, which regions of the input space need more data?
- Is the selected parametric model suitable?
- What is the conditioning of a data-fitting problem?
- Is data-based mathematical modeling promising for the given task?

The developed MC-HARP environment can support the diverse needs of the scientific and engineering community. It has the versatility to develop and verify parametric and nonparametric mathematical models and also global and local approximate mappings. Furthermore, it establishes an environment for unifying existing mathematical modeling techniques in statistics, approximation theory, information theory, system identification, and neural networks.

7.1 Summary

In Chapter One, we described the desirable characteristics of a data-based mathematical modeling tool for diverse needs of engineering applications. We justified the need for such a numerical tool by describing its engineering applications especially for areas like mathematical modeling, information processing and knowledge representation, pattern recognition and classification, fault detection and diagnosis, numerical approximation, and control.

In Chapter Two, we presented a robust method for approximating multivariate mappings, based on the concept of *hierarchical adaptive random partitioning* (HARP). The basic nature of the method is local approximation. The input domain is partitioned into subdomains and independent local approximations are built for each subdomain. The concepts of inverse image, partitioning function, and characteristic function are used to represent a HARP approximation by simple operations of summation, multiplication, and composition of univariate functions. The HARP partitioning has the flexibility to adapt to the behavior of the data and is driven by approximation errors. More subdomains are generated where the data have complex behavior. The subdomains are randomly divided and their training processes are independent of one another, so these computations are efficient and well suited to parallel processing.

We used fuzzy subdomains to enforce continuity of the mapping constructed by the HARP and to smooth the mismatching of the local approximations in the neighboring subdomains. The HARP method gives rise to many equally plausible solutions to a data fitting problem. We applied a Monte Carlo strategy on top of the HARP algorithm and developed a new method for creating a data-based mapping approximation called
the *MC-HARP method*. The MC-HARP approximation is the mean average of a sample of HARP approximations. For each input pattern, the MC-HARP method can compute a confidence index for the predicted output.

As the amount of data increases, the MC-HARP approximation converges to the actual mapping more uniformly and with less localized disturbances than any individual HARP approximation. The constructed MC-HARP approximation shows good generalization because it has the flexibility of the local approximations to adapt to complex, nonhomogeneous functional behavior and the smoothness of the global approximations to capture the global features of the data. The MC-HARP method preserves all the main characteristics of the HARP algorithm, namely structural self-organization, fast learning, and automatic processing. The independence of subdomain approximations and of HARP approximations make the MC-HARP method highly parallelizable.

In Chapter Three, we studied the performance of an MC-HARP approximation through numerical simulations. We showed that by increasing the sample size for the MC-HARP method, the performance and complexity measures of an MC-HARP approximation converges, a feature inherited from the Monte Carlo process. We set the minimum sample size for MC-HARP to be the sample size for which the deviation measure has converged. We declared that the MC-HARP approximation is consistent and that its complexity saturates for large data sets. In other words, by increasing the amount of data, the approximation converges to the actual mapping and its complexity increases at a rate smaller than the rate at which new data are introduced. We suggested the use of simple subdomain approximations unless one can demonstrate that a more complex subdomain approximation would improve the performance or reduce the complexity of the constructed MC-HARP approximation, or if prior knowledge about the mapping approximation problem suggests a more complex subdomain approximation.

We showed that the tolerance value for the acceptability of the subdomain training process is the parameter that controls the complexity of an approximation-error-driven partitioning processes like HARP. Consequently, HARP partitions, HARP approximations, and MC-HARP approximations are functions of the selected tolerance value. By increasing the tolerance value, the complexity of an MC-HARP approximation decreases and its approximation error increases. Furthermore, We showed that the boundary fuzziness of HARP approximations improves the performance of an MC-HARP approximation for regions of the input domain where the actual mapping has complex behavior in comparison to the selected subdomain approximation or the training data points are sparse.

We studied the behavior of an MC-HARP approximation with respect to the dimension of the input domain. We demonstrated that the rate of convergence for an MC-HARP approximation is independent of the dimensionality of data and its ultimate rate of convergence is \( o(1/\sqrt{N}) \), where \( N \) is the number of training data points. Numerical simulations indicated that the superposition of HARP approximations through MC-HARP improves the approximation accuracy and the MC-HARP approximation converges to the actual mapping more uniformly than do HARP approximations.

We showed that HARP and MC-HARP approximations can be implemented on neural hardwares. An MC-HARP approximation can be modeled as a modular neural network whose basic module is a HARP neural network. We demonstrated that HARP and MC-HARP neural networks need only standard sigma units. Fur-
thermore, we demonstrated that MC-HARP constitutes an environment for simultaneously building and training mapping neural networks. The HARP and MC-HARP training processes are highly parallelizable and can exhibit good speed up on multiprocessor computers. A HARP neural network, besides the fundamental components of connectivity pattern, propagation rule, and learning rule, also has a growing process. The subdomain partitioning stage of a HARP training process grows the constructed network and allocates new partitioning and approximation units. The weight updating of a HARP training process is localized. A HARP neural network only adjusts the weights corresponding to those subdomain approximations whose subdomains contain the presented training patterns.

In Chapter Four, we used numerical simulations to study performance of the MC-HARP method for noisy data. We showed that the tolerance value for the termination criterion for the subdomain training process of MC-HARP controls the distribution, size, and number of subdomains in HARP partitions. The model selection problem for an MC-HARP approximation was defined to be, the selection of the optimal tolerance such that its corresponding MC-HARP approximation has the minimum approximation risk in the family of approximations built by MC-HARP with different complexities.

Performance indices were defined to investigate the complexity-dependent accuracy of an MC-HARP approximation for different noise amplitudes and amounts of data. General trends in performance of constructed MC-HARP approximations were extracted to establish a framework for performance analysis of MC-HARP. The performance of an MC-HARP approximation with low complexity, high complexity, and optimal complexity were investigated. We showed that there exists an optimal tolerance value corresponding to an approximation with the optimal complexity and lowest approximation error. We presented that the optimal tolerance is equal to zero for noise-free data and is greater than zero for noisy data. The value of the optimal tolerance is a function of the amplitude of noise added to the data. We showed that HARP and MC-HARP approximations are consistent only for the unique value of the optimal tolerance.

In Chapter Five, we proposed a new model selection criterion using the deviation measure computed by MC-HARP. The proposed MC-HARP model selection is based on the minimization of the deviation measure in the limit with respect to the amount of data over the entire input domain. The minimization in the limit and consideration of the entire input domain are the two main characteristics of the proposed model selection technique that distinguish it from current techniques which all use a straight minimization over the set of training data points or a subset of it.

We expressed the MC-HARP philosophy for performance estimation of data-based approximate mappings and illustrated its advantages to sampling-based techniques. Unlike the sampling-based techniques for performance estimation, the MC-HARP method considers the spatial neighboring relations among data points and uses them not only for building its approximation but also for estimating a pointwise, approximation deviation measure. The MC-HARP deviation measure represents how strong the nearby data for a point dictates its predicted output.

We defined the deviation measure of an MC-HARP approximation for the optimal tolerance value as a quantitative measure for the approximation confidence, approximation accuracy, and the adequacy of data. We established a novel MC-HARP framework for classifying data-fitting problems with respect to the quality-
quantity conditioning of their data sets. The MC-HARP framework suggests a quality-quantity map for a data set and a quantitative measure for its conditioning. The proposed framework is able to detect an ill-conditioned data-fitting problem and to warn the experimenter about the unreliable performance of the constructed mathematical model. We introduced the concept of measurability for the conditioning of a data set. We developed a framework of rules based on the features in the MC-HARP computable performance curves to classify a data set with respect to its conditioning. Furthermore, the MC-HARP framework presents a series of rules for bounding the optimal tolerance value and selecting the most suitable tolerance value for a data set that is not ill-conditioned. We used a set of real data to illustrate the application of the proposed MC-HARP model selection technique and framework for classifying nonparametric, data-fitting problems.

In Chapter Six, we showed a data-based mathematical model can be built for the long term performance of a pavement. We used the developed MC-HARP method with the MC-HARP model selection technique and framework for classifying data sets to build an empirical model for pavement performance using the data of the AASHO Road Test. We demonstrated the superior performance of the MC-HARP model to the AASHO model currently used for pavement design. We concluded that a local approximation like an MC-HARP approximation can model the pavement performance data for the entire input domain better than a global approximation like the AASHO formula.

By selecting a suitable complexity, we simplified the constructed MC-HARP approximation and improved its performance for test data points without significantly degrading its performance for the training set. We used MC-HARP to construct an MC-HARP approximation whose parametric subdomain approximation is the AASHO model. We showed that the performance of an MC-HARP approximation can be improved by selecting a subdomain approximation that accounts for prior knowledge of the physics of the pavement performance modeling problem by selecting a specific functional form for the parametric subdomain approximation. The applicability of the MC-HARP method for nonlinear subdomain approximations was shown by using the highly nonlinear AASHO subdomain approximation. Finally, we illustrated how MC-HARP can be used to verify an existing parametric mathematical model.
APPENDIX A
Neural Networks

Recent progress in computer and other high technology industries has made the gathering of information and data easier. In many fields, enormous amounts of information exist or can easily be generated. The need for a fast data processor and knowledge representer is evident. Neurocomputing and parallel distributed information processing systems like neural networks represent a promising approach to filling this need. Neural networks introduce a fundamentally new approach to information processing that does not require the development of algorithms or rules. The architecture of a neural network is inspired by the structure of the brain, as the name implies, and represents the first important alternative to programmable computers. The potential of neural networks remains largely unrealized. The large number of research and application papers appearing in conferences concerning neural networks (ANNIE 1991 through 1993; EURASIP 1990; ICANN 1991 through 1994; ICNN 1987, 1988, and 1993; IEE/ICANN 1989, 1991, 1993; ICNN 1989 through 1993; INNC 1990; IWANN 1991 and 1993; NEURONET 1990 and 1993; NIPS 1989 through 1993; WCNN 1993 and 1994; WNN-AIND 1991 through 1993) is indicative of the considerable interest in this area of research.

Neural networks use a parallel, distributed processing structure constructed from a set of simple, interconnected processors. This processing system can organize itself to represent the knowledge contained in the data in an optimal way. Besides this self-organization feature, neural networks possess a parallel structure that has a powerful potential for creative hardware implementations of massively parallel processors. Neural networks can be applied to a broad range of problems in different fields, including civil and mechanical engineering, such as pattern recognition, data analysis, sensor processing, classification, and control where algorithms and rules are not known.

The purpose of this appendix is to briefly introduce the concept of neural networks, elucidating their structure, and describing how they process information and self-organize.

A.1 What Is a Neural Network?

The architecture of neural networks is inspired by the network of nerve cells (neurons) in the brain that provide the functionality of the brain. Although, our current knowledge of the brain is limited, neural networks attempt to simulate what goes on in the nervous system, with the hope of achieving some of the brain's powerful capabilities such as vision, learning, remembering, and thinking. The human nervous system consists of billions of neurons, richly interconnected, doing relatively simple computations without the benefit of a programmer.
The fundamental cellular unit of the nervous system is the neuron. The input area of the neuron is a set of branching fibers called dendrites and its output area is a long branching fiber called an axon. The connecting point between a neuron's axon and another neuron's dendrite is called the synapse. Each neuron is a simple processing unit that receives a series of impulses at its dendritic area. If the total potential arriving at the cell body within a short period surpasses a threshold, then the probability that the target neuron will fire an impulse down its axon is very high. The impulse is transmitted across the synapse by chemical reactions and the amount of signal transferred depends on the amount of chemicals released by the axon and received by the dendrites. This synoptic efficiency (strength) changes as the brain learns.

A neural network is an interconnected parallel distributed network of many simple processing units (Fig. A.1). Each unit can process a local memory and is capable of performing a few simple computations such as summation or comparison to a threshold value. Units are connected via unidirectional signal (information) channels called connections and have a single output connection that fans out into many collateral connections which carry the same signal. Each connection has an associated, adjustable weight (strength) that modifies the transmitted signal.

In a neural network, the processing unit is analogous to the neuron. As shown in Fig. A.2, a unit receives multiple inputs from other units through its input connections. Each interconnection has an associated weight given by \( w_{1j}, w_{2j}, \ldots, w_{nj} \), that simulates the strength of the biological synapse. The processing unit combines the inputs and computes its net input usually by a simple weighted summation. Then, this unit computes its output by applying its output function to the net input. This output function can be a threshold function which only passes information if the net input reaches a certain value or it can be a continuous function. Subsequently, the calculated output is sent along the output connection of the processing unit to the other units to which it is connected. The output signal of a processing unit can be of any mathematical type desired, including binary, integer, real, complex or fuzzy numbers. The processing inside each unit can be arbitrarily defined with the constraint that it must be completely local, which means the computations must depend only on the
received inputs, the unit's incoming connection weights, and values stored in the unit's local memory (depending on the hardware, the weights can also be stored in the local memory).

The processing units are normally arranged in layers as shown in Fig. A.1. There are typically two layers with connections to the outside word. The layer of units that receives signals from an external source is the input layer and the output layer is the group of units whose outputs are sent to an external receiver. Layers distinct from the input and output layers are called hidden layers. Computation starts when the network is presented with an input pattern. Then the input units produce outputs that are transmitted to other units and thus the propagation process begins. Like the input pattern, the response of the neural computation is an output pattern, the output of the output layer.

One of the characteristics of the neural networks that has excited many people is their capability to self-organize or learn. A neural network is not programmed like a computer by giving it an explicit sequence of instructions describing how the system should proceed. Rather, the network adapts itself to capture concepts just by looking at the available examples (data). One trains a neural computing system by presenting a set of example stimuli associated with a concept, to achieve a particular self-organization goal. Neural networks not only learn the presented data (training cases) but also are able to generalize the rules from the training cases and apply these rules to new cases. Learning and generalization are two basic characteristics of neural networks and both of them originate from the property that a neural network dynamically changes its connection weights through self-organization. A neural network records what it has been taught by assigning weights to the connections between units. Learning is the process of self-adaptation at the processing level. The weights are automatically modified to achieve specific results, eliminating the need for writing a specific algorithm for each problem. As in biological systems, the weight (strength) of the connections (synapses) can change in response to the strength of the inputs and the type of the output function used by the units. These connection weights cannot be set manually even for simple problems. Therefore, a network must undergo a training procedure. If the output pattern computed by the network is close to the expected value for a given input pattern then the computation performed by the network is correct. A reasonable measure of error in a neural network is the difference between observed and expected output patterns. The error can be used to adjust the weights so that the next time that the same input pattern is presented, the network will come a bit closer to producing the desired response. This kind of training process, in which the network is presented with the

Fig. A.2 Schematic processing unit
desired output for each input pattern, is referred to as \textit{supervised training}. If no expected output is shown then the training is \textit{unsupervised}. During such self-organization a network divides input patterns into clusters in accord with their similarities. Regardless of the type of learning, an essential characteristic of any network is its \textit{learning rule}, which specifies how weights are adjusted in response to a training example. Generally, learning is an iterative process and one needs to present the set of training examples to the network many times.

\subsection*{A.2 Fundamental Components of a Neural Network}

There are many kinds of neural networks, including the perceptron, adaline and madaline, back-propagation, bi-directional associative memory, Boltzmann machine, counter-propagation, adaptive resonance theory, Hopfield network, Hamming network, probabilistic neural network, self-organizing feature maps, and learning vector quantization (Zurada 1992, Wasserman 1989, Carpenter 1990, Hecht-Nielsen 1990, Dayhoff 1990, and Klimasauskas 1989). These networks differ from each other in their fundamental components, namely the pattern of connectivity, propagation rule, and learning rule. We present an overview of these basic aspects in the following subsections.

\subsection*{A.2.1 Pattern of Connectivity}

The processing units are usually arranged in layers and the units in one layer are generally connected to many units in other layers. Sometimes units are connected to the units in their own layer or even to themselves. One type of neural networks is called a \textit{feedforward network} wherein propagation of computation takes place in a forward manner from input layer to output layer without any feedback. A feedforward neural network is shown schematically in Fig. A.3. This kind of network is very popular due to its relative simplicity, stability, and fast response. A network whose pattern of connectivity allows information to flow both forward and backward is called a \textit{feedback network}. In such a network, units are connected to one another across layers or within layers (see Fig. A.1) and computation continues until some convergence criteria are met.
A.2.2 Propagation Rule

Processing units propagate information through the network. As shown in Fig. A.2, these units produce their output by two main operations, computing net input and applying an output function. Many different kinds of processing units have been introduced and differ from each other in these main operations. In this section, we explain some of the currently popular ways of doing these computations.

First, for computing the net input, the most popular rule is a simple weighted summation of inputs, described mathematically as follows

\[ \Omega_j = \sum_{i \in I_j} w_{ij} o_i \]  

(A.1)

where \( \Omega_j \) is the net input to the \( j \)th unit, \( o_i \) is the output of the \( i \)th unit, \( w_{ij} \) is the weight of the connection from the \( i \)th unit to the \( j \)th unit, and \( I_j \) is an index set containing unit numbers of units sending their outputs to the \( j \)th unit. Eqn. (A.1) is simply the scalar product of the input vector and the weight vector of connections from other units coming into the \( j \)th unit. The maximum net input is achieved if these vectors point in the same direction.

A generalized form of the weighted summation can be defined as

\[ \Omega_j = \sum_{i \in I_j} w_{ij} \prod_k o_{ik} \]  

(A.2)

where the multiplication over index \( k \) is computed for the collection of subsets of \( I_j \) whose outputs are assigned to be multiplied by each other before they are modified by the weights. A unit computing its net input based on Eqn. (A.2) is called a \textit{sigma-pi} unit.

In feedback networks, Eqn. (A.1) can take the form as

\[ \Omega_j = \Omega_j^{old} + \sum_{i \in I_j} w_{ij} o_i \]  

(A.3)

where \( \Omega_j^{old} \) is the prior net input of the \( j \)th unit during iterative signal propagation of a network. \( \Omega_j^{old} \) can be stored in the unit’s local memory (Fig. A.2).

The net input can be the maximum of the weighted inputs, that is,

\[ \Omega_j = \max_{i \in I_j}(w_{ij} o_i) \]  

(A.4)

This type of computation is used when competition among inputs is required with a winner selected.

Sometimes units can act like multipliers and the net input is a weighted product as follows

\[ \Omega_j = \prod_{i \in I_j} w_{ij} o_i \]  

(A.5)

In unsupervised learning, a network self-organizes in order to reduce the difference between its weights and given input patterns. Here, the output of a unit depends on how close the input vector and the weight vector
are. In this case, the net input indicates the distance between these vectors. Besides the weighted sum, the Manhattan distance is also used to calculate the closeness. This distance is defined as

\[ \Omega = \sum_{i \in I} |o_i - w_i| \]  \hspace{1cm} (A.6)

The input and weight vectors are normalized to calculate the distance. In the literature one can find other variations and combinations of these propagation rules.

After a processing unit computes its net input, it imposes a typically non-linear, univariate output function on its net input to compute its output. Associated with each unit is an output function \( f \), and the mathematical relationship between the net input \( \Omega \) and output value \( o \) for any unit can be described as

\[ o = f(\Omega) \]  \hspace{1cm} (A.7)

Many kinds of output functions have been presented in the literature, inspired either from biological or mathematical bases. The simplest output function is the identity (linear) function, defined as

\[ o = \Omega \]  \hspace{1cm} (A.8)

The most popular output function is the sigmoid function which is a bounded, differentiable, non-decreasing function defined for all real numbers (see Fig. A.4). A typical form of a sigmoid function, whose range is \((0,1)\), is defined as

\[ o = \frac{1}{1 + e^{-\Omega}} \]  \hspace{1cm} (A.9)

Other popular forms of sigmoidal output functions are hyperbolic tangent, tangent inverse, and \( \frac{\Omega}{1 + |\Omega|} \). The output function can be nonmonotonic, e.g. a harmonic function or piecewise defined, e.g. a step function with a threshold value of \( a \) and a range equal to \( \{b, c\} \) (see Fig. A.4).
\[ o = \begin{cases} c & \text{if } \Omega > a \\ b & \text{if } \Omega \leq a \end{cases} \]  
(A.10)

The perceptron type of output function is a piecewise linear function with a lower limit of zero (Fig. A.4),

\[ o = \begin{cases} \Omega & \text{if } \Omega > 0 \\ 0 & \text{if } \Omega \leq 0 \end{cases} \]  
(A.11)

As shown in Fig. A.4, the ramp function is another type of output function that is piecewise linear and bounded

\[ o = \begin{cases} c & \text{if } \Omega \leq a \\ \frac{\Omega(d - c) + bc - ad}{b - a} & \text{if } a < \Omega < b \\ d & \text{if } \Omega \geq b \end{cases} \]  
(A.12)

### A.2.3 Learning Rule

The learning rule defines how the network is adjusted in response to the presentation of training cases. The connection weights are automatically modified to achieve a self-organization through a training process. In each iteration of the training process increment values are computed for all connection weights of a network through a learning rule and then the connection weights are modified by simply adding their computed increments to their current values. All rules for modifying the weights are a variant of Hebbian learning inspired by biological nervous systems. The strength of the synapse between two neurons is change based on the activations level of these cells (Hebb 1949). The simplest case of Hebbian learning can be stated as

\[ \Delta w_{ij} = \xi o_i o_j \]  
(A.13)

where \( o_i \) is the output of the \( i \)th unit sent to the \( j \)th unit, \( \xi \) is the learning rate, and \( \Delta w_{ij} \) is the increment for the weight of the connection from the \( i \)th unit to the \( j \)th unit. If the \( j \)th unit is an output unit then \( o_j \) is set to be the desired target value \( t_j \). The learning rate \( \xi \) represents the degree by which the weights are adjusted when both units are excited and is usually set to one.

Another form of learning rule is called the adaline rule (Widrow 1962) and is used to adjust the weights in the adaline network as follows

\[ \Delta w_{ij} = \frac{1}{n}(y_j - \Omega_j) o_i \]  
(A.14)

where \( \Omega_j \) and \( y_j \) are the net input and the target value of the \( j \)th unit, respectively and \( n \) is the number of inputs \( o_i \) to the \( j \)th unit. This learning rule causes the weights to adapt even if the output is correct and this behavior may cause the divergence of the learning process.

A common learning rule is called the delta rule or generalized delta rule (Rumelhart, et al. 1986) and takes the form
\[ \Delta w_{ij} = \xi \delta_j + \alpha \Delta w_{ij}^{old} \]  

(A.15)

where \( \delta_j \) is the error of the \( j \)th unit, \( \Delta w_{ij}^{old} \) is the prior weight increment during the last learning iteration, and \( \xi \) and \( \alpha \) are the learning constants. \( \Delta w_{ij}^{old} \) is called the momentum term and is used to prevent oscillation during learning. The delta rule will be discussed in more mathematical detail in the next section on backpropagation neural networks. The error \( \delta_j \) for the output units is computed as the difference between the target output \( y_j \) and the actual output \( o_j \). For the hidden units the error term is computed from the errors at the output layer based on a series of chained computations which backpropagate the error inside the network. The learning constants are chosen between zero and one.

For unsupervised learning, the weight updating should be only based on the input values and intend to make the weight vector and the input vector as close as possible. The learning process requires that the processing units compete, which means that a unit with the lowest distance (such as Euclidean or Manhattan distance) between its input and weight vectors or in case of normalized weights, a unit with the highest net input is the winner. The connection weights of the winner unit \( j \) to the input layer are updated as follows

\[ \Delta w_{ij} = \xi (o_i - w_{ij}) \]  

(A.16)

and the weights of the rest of connections do not change (Kohonen 1984). Equation (A.16) changes a weight vector such that it more closely approximates the input vector that caused the unit to win and more likely the unit will be the winner for the other input vectors near the current input vector. In other words, the weight vector of each unit becomes the average of all input vectors that cause it to win. The learning constant \( \xi \) is typically set to a small value (0.2 or less) and for very large training sets, even smaller values are recommended.

There are other learning rules developed for the specific networks such as Hopfield (Hopfield 1982, 1983, and 1984), perceptron (Rosenblatt 1988 and 1990), Boltzmann (Hinton, et al. 1983 and 1984), adaptive resonance theory (Carpenter and Grossberg 1987a and 1987b), and probabilistic neural networks (Specht 1988 and 1990). Some of these rules, like the Hebbian rule, try to model the adaptation of the biological system in much more complex way and the others utilize optimization and statistical methods to make the learning process faster and more robust.

A.3 Backpropagation Neural Networks

The backpropagation network is the most widely used neural network. The reason for this popularity is its solid mathematical basis and production of a number of successful applications. Backpropagation gives a way to adjust weights in feedforward networks with many layers. The power of backpropagation lies in its ability to train hidden layers using a more sophisticated learning rule and thereby escapes the restricted capabilities of networks without hidden layer such as linear separability of the training patterns (Minsky and Morgan 1969). Backpropagation can be used for any problem that requires pattern mapping from an input pattern to an associated output pattern. Backpropagation has a broad range of applications from military pattern rec-
ognition to medical diagnosis, and from credit application scoring to image compression. It was originally introduced by Paul Werbos (Werbos 1974) and then independently reinvented by David Parker (Parker 1985, 1986, and 1987). Arthur Bryson and Yu-Chi Ho (Bryson and Ho 1969) presented a mathematically similar recursive algorithm for control applications. Backpropagation was popularized by Rumelhart and other members of the PDP group (Rumelhart, et al. 1986). Their work introduced the potential of neurocomputing and generated the widespread interest in backpropagation.

Typically in the backpropagation network, the processing units are arranged in layers and the network has a feedforward pattern of connectivity. As shown in Fig. A.3, the layers are fully interconnected and each unit can be connected to every unit in the layers above its own layer, but there are no connections within a layer. The first layer of the network is the input layer and the last layer is the output layer. A backpropagation network must have at least these two layers. A processing unit computes its net input by a weighted summation as in Eqn. (A.1) or in a more generalized case by the sigma-pi rule in Eqn. (A.2). The output function is usually a sigmoid function. The output of each unit is generally a real number bounded within the range of the output function. After presenting an input pattern to the input units, the response of the network is produced by a series of forward propagations from the input layer, through each hidden layer, to the output layer of units.

A backpropagation neural network intends to approximately construct a bounded mapping \( F : \mathcal{D} \subset \mathbb{R}^n \rightarrow \mathbb{R}^m \) from an \( n \) dimensional set \( \mathcal{D} \) to an \( m \) dimensional space for a set of training examples \( \{(x_q, y_q), q=1, \ldots, N : x_q \in \mathcal{D}, y_q \in \mathbb{R}^m\} \). The network is trained through a supervised learning process. One presents the network with pairs of patterns; an input pattern associated with a target output pattern. In each presentation, weights are adjusted to decrease the distance between the response of the network and the desired output. The training set is presented to the network many times and after training is stopped, the performance of the network is tested.

A.3.1 Learning Process

The learning rule for a backpropagation network is the generalized delta rule defined in Eqn. (A.15). The training process is iterative. Each iteration has a forward propagating step followed by a backward propagating step and is called a cycle. In the forward propagating step, the network is presented with an input pattern at its input layer and then propagates the activation forward through the hidden layers. The output layer produces the response of the network. During this step, the output of each unit \( o_i \) is stored to use in the learning rule. The backward propagating step starts by computing the error between the network's response and the expected output pattern. Then the output units backpropagate their errors to calculate \( \delta_j \) values for hidden units. Then the connection weights are modified using the generalized delta rule.

The basic idea of the generalized delta rule is to perform a steepest descent of the total error which is the squares of the errors between the network's outputs and the desired outputs summed over the output units and all training set (Rumelhart, et al. 1986). The error \( E \) is set to be the sum of errors for each training pattern and takes the form

\[ E = \sum_{q} \sum_{i} (o_i - y_i)^2 \]
\[ E = \sum_{q=1}^{N} E_q \]  

(A.17)

where \( N \) is the number of training patterns and \( E_q \) is the measure of error for the \( q \)th training pattern defined as

\[ E_q = \frac{1}{2} \sum_{j=1}^{m} [y_j^q - F_j(x_q; w)]^2 \]  

(A.18)

where \( m \) is the number of output units, \( y_j^q \) is the target value for the \( j \)th output unit for the \( q \)th training pattern and \( F_j \) is the network computed output of its \( j \)th output unit. The response vector of the network \( \{F_j: j=1,\ldots,m\} \) is a function of the input vector \( x_q \) and the connection weights \( w \). The training process adjusts the connection weights such that the total error \( E \) is minimized. The generalized delta rule represents steepest descent method for minimizing the total error \( E \) as a function of connection weights \( w \). The step length in each minimization iteration is kept constant and is called the learning rate. The generalized delta rule without the momentum term can be written as follows

\[ \Delta_q w_{ij} = \xi \delta_j^q o_j^q \]  

(A.19)

where \( \Delta_q w_{ij} \) is the weight increment for the weight of the connection from the \( i \)th unit to the \( j \)th unit after presenting the \( q \)th training pattern and \( \xi \) is the learning rate. If the \( j \)th unit is an output unit, then the \( \delta_j^q \) value is computed as

\[ \delta_j^q = -(y_j^q - o_j^q) \cdot \frac{df_j}{d\omega_j^q} \]  

(A.20)

where \( f_j \) is the output function for the \( j \)th unit and \( \omega_j \) is its net input. For units in each hidden layer, \( \delta \) values can be computed from the \( \delta \) values of the units in the higher layers as follows

\[ \delta_j^q = \frac{df_j}{d\omega_j^q} \sum_k w_{jk} \delta_k^q \]  

(A.21)

where index \( k \) sweeps all the units to which the \( j \)th unit sends its output. Equation (A.21) shows that the \( \delta \) value of a hidden unit is a weighted summation of the \( \delta \) values of the units in the higher layers that it influences. Figure A.5 illustrates different steps in the learning process of the backpropagation network. The backward propagation of \( \delta \) values from the output layer is similar to the forward propagation of activations from the input layer.

Two different strategies are in common for updating the weights. In the first approach, which is called online or continuous updating, weights are adjusted after each training pattern is presented. In the second approach, weight changes are accumulated over several training presentations (epoch) and applied at once. This approach is called batch or periodic updating. The batch method applies the true gradient of the total error \( E \) if the size of the epoch is equal to the number of training patterns. On the other hand, the online approach departs from the true gradient descent in \( E \). However, by providing a sufficiently small learning rate,
A.3.2 Building and Training a Backpropagation Network

Like any specialized data processing tool, one must verify that the backpropagation network is a suitable tool for the problem. The backpropagation network is proper for pattern association and mapping type of problems for which a large number of training examples exist. After the verification step, there are four main steps in the development of a backpropagation neural network: (1) the selection of the architecture; (2) splitting the available data into training and testing sets; (3) the training of the network with the training set; and (4) testing the generalization of the network with the testing set.

The architecture of the network is selected by determining the number of hidden layers, the number of units in each hidden layer, the output function of each unit, and the connectivity pattern among units. Currently, there are no good rules for the selection of an architecture. However, experience with certain architectures has given some insight into what a good architecture might be. A bad architecture fails to reduce the error for all training cases or performs poorly over the testing set. The network must be complex enough to capture the behavior of the data. However, the number of possible architectures grows geometrically with the number of parameters that must be specified. Consequently, finding a suitable architecture can require an enormous amount of computation, even for small problems. (One of our main goals in the present work is to develop a procedure for determining the network's architecture.)

Typically, the development of a backpropagation network requires both a training set and a testing set. The training set is used to train the network and should embody the various features that the network is likely
to encounter. In other words, the training cases should be rich and cover the whole application domain of the network. Generally, both sets are taken from real data, although sometimes simulated data are used as well. In a typical application the available data is randomly divided into training cases and test cases. Random splitting works well where a large number of data are available. For small data sets, appropriately perturbing, combining, or adding noise to the existing data may be used to generate additional cases. The training and testing sets should cover approximately the same region in the domain of the network and have similar statistical properties.

The training process is started by initializing the connection weights with random values and then training cases are presented to the network. The connection weights are adjusted according to the generalized delta rule described in previous section. The training process stops when the convergence index is below a pre-specified value. The convergence index can be the maximum of the square error $E_q$ in Eqn. (A.18) for all output units or the maximum error of the output units for all training patterns. The generalized delta rule is a gradient descent method and hence requires many iterations to converge and sometimes gets stuck at a local minimum. Many techniques have been developed to speed up the training process of the backpropagation and some of them are discussed in the next section. The value of the learning rate $\xi$ affects the speed of convergence. If $\xi$ is very large, instability in the network and oscillations in the learning process can occur. Learning rates that are too small can lead to excessively slow learning, but with little oscillation. The momentum parameter $\alpha$ in Eqn. (A.15) is an inertial parameter that damps out local oscillations and provides additional speed up (Rumelhart, et al. 1986). The learning constants $\xi$ and $\alpha$ usually lie somewhere between zero and one. The choice of the best values for these constants is especially hard when the problem being addressed is very large, involving a lot of cycles for a single sweep through the training set. As the learning process continues, decreasing the learning constants can sometimes produce better convergence.

When a local minimum is encountered, a number of techniques can be used to help the network not to get stuck at that local minimum. For example, changing the learning parameters or the number of hidden units (Ash 1989, Fahlman and Lebiere 1990, and Karnin 1990) and adding perturbations to the weights may cause the network to avoid the local minimum. If the learning process gets stuck at a local minimum, adding small random values (perturbations) to the weights often allows the network to escape from the basin of the local minimum and then training may proceed in a new direction.

After the training process, the mapping learned by the network is embodied in the connection weights and the network can be used to produce response for any input pattern. The last step in the development of a backpropagation network is to test the generalization of the trained network. Generalization is the ability of the network to produce correct responses for test cases which were not explicitly presented to the network. The testing step is performed to see how well the network has captured the hidden features in the training set. The generalization or prediction capability of the network is measured by its performance over the testing set. The percentage of the correct responses and the total error for all test cases are some of indices to measure the generalization. If the network has a poor generalization, its architecture should be changed. However, if there are a few test cases for which the network has large amount of error, those cases can be added to the training set.
A.3.3 Speeding up Techniques for Backpropagation

Backpropagation learning is a simple gradient optimization procedure and therefore its convergence is very sensitive to the shape of the total error hypersurface. An important drawback of backpropagation is the slowness of convergence when the error function presents flat hypersurfaces and ravines. Experience has shown that the backpropagation error function has extensive flat areas and troughs that have very little slope (Hecht-Nielsen 1990). In these areas, the learning process spends a lot of iterations before a significant drop in error occurs. The problem of speeding up backpropagation has been approached in two ways: methods incorporating higher order information about the surface into the learning algorithm and the heuristic techniques.

The higher order approach tries to use numerical techniques for nonlinear optimization. These techniques have been extensively researched and their stability and convergence rate have been investigated computationally (Luenberger 1989 and Fletcher 1980). The general goal of this approach is to provide a faster descent to the bottom of the error surface. Gradient methods such as conjugate gradient, Newton, and quasi-Newton methods have been applied to guide the learning process to a suitable minimum of the error function (Watrous 1987, Himmelblau 1990, and Battiti 1990). In these methods weight updating at the kth iteration (cycle) of learning process has the following form

\[ w_{k+1} = w_k + \xi d_k \]  

where \( w_k \) is the vector of weights, and \( d_k \) is the search direction. The step length \( \xi \) can be a constant value or determined by a line search process that forces the optimization method to descend. In the generalized delta rule, the search direction is given by the negative gradient of the total error function and \( \xi \) is the fixed learning rate. In the conjugate gradient algorithm, the search direction is defined as

\[ d_k = -g_k + \alpha \Delta w_{k-1} + \beta q_k \]  

where

\[ g_k = \nabla wE(w_k) \]
\[ q_k = g_k - g_{k-1} \]
\[ \Delta w_{k-1} = w_k - w_{k-1} \]  

\[ \beta = \frac{\Delta w_{k-1} \cdot g_k}{\Delta w_{k-1} \cdot q_k} \]
\[ \alpha = -\left[ 1 + \frac{q_k^T q_k}{\Delta w_{k-1}^T q_k} \right] \beta + \frac{g_k^T q_k}{\Delta w_{k-1}^T q_k} \]

Equation (A.23) has the same form as the delta rule in Eqn. (A.15). The search direction \( d_k \) is a combination of the gradient of the error function \( g_k \), momentum term \( \Delta w_{k-1} \), and gradient increment \( q_k \). In the conjugate gradient algorithm, momentum coefficient \( \alpha \) and constant \( \beta \) are computed automatically in each iteration.

In the Newton method, the search direction is computed as follows
where $g_k$ has been defined in Eqn. (A.24) and $H_k$ is the Hessian matrix of the error function computed at $w_k$. The Newton method achieves rapid convergence near the minima where objective function is more quadratic but it needs higher order derivatives of the error function and matrix inversion. To overcome these drawbacks, quasi-Newton methods can be used. These methods update the inverse of the Hessian matrix with a lower rank matrix built of the gradient and search direction vectors. The BFGS algorithm from the quasi-Newton family updates the inverse of the Hessian matrix as follows

$$
H_k^{-1} = H_{k-1}^{-1} + \left[ 1 + \frac{q_k^T H_{k-1}^{-1} q_k}{\Delta w_{k-1}^T H_{k-1}^{-1} q_k} \right] \frac{\Delta w_{k-1} q_k^T H_{k-1}^{-1}}{\Delta w_{k-1}^T q_k} - \frac{\Delta w_{k-1} q_k^T H_{k-1}^{-1}}{\Delta w_{k-1}^T q_k} \quad (A.26)
$$

where vectors $\Delta w_{k-1}$ and $q_k$ have been defined in Eqn. (A.24). Therefore quasi-Newton methods like the conjugate gradient method require only the gradient of the error function and compute the inverse of the Hessian by rank-two updating, not by direct inversion. Also, the computational complexity of the Newton method can be reduced by using an approximation of the Hessian matrix in Eqn. (A.25). This approximated Hessian matrix can be the diagonal part of the exact Hessian matrix (Becker, et al. 1988) or computed by the Gauss-Newton algorithm for the least squares method.

These higher order methods have had their most prominent success in second-order backpropagation (Becker and Le Cun 1988, Ricotti, et al. 1988, Watrous 1987, and Parker 1987). Although these learning methods are certainly useful and have the advantage of a faster adaptation, because of higher computational complexity, implementation difficulties, ill conditioning of the Hessian matrix for feedforward neural networks, storage requirements, and the relatively small real time speed up that is achieved, they have not been widely used. Another drawback of these higher order methods is their nonlocality. A local learning technique modifies each estimate of a connection weight based solely on information about that weight and performs local computations. In other words, the learning rule for a local learning process does not depend on factors computed using information in the entire network. For example in the Newton method, computing the Hessian matrix is a nonlocal process which requires the computation of all units and therefore in the learning rule, Eqns. (A.22) and (A.25), weight updating for each connection is completely correlated to weight updating for the other connection weights and information in all units. Local learning methods are more biologically accepted and easier to implement in parallel architectures than nonlocal techniques.

The other group of methods for acceleration of backpropagation are based on intuitive heuristics and try to produce empirically acceptable results through local computations. One of the earliest methods in this group is the use of the momentum term in the generalized delta rule (Rumelhart, et al. 1986). It modifies the steepest descent formula to be

$$
\Delta w_{ij}^k = -\xi \frac{\partial E}{\partial w_{ij}} + \alpha \Delta w_{ij}^{k-1} \quad (A.27)
$$
where $\xi$ and $\alpha$ are learning rate and momentum constant, respectively and $k$ is the iteration counter for the training process. Momentum has been so successful that it is now an integral part of almost all backpropagation networks. Stornetta and Huberman (1987) modify the sigmoid output function such that the activation range of all units in the network are symmetric about zero ranging from $-1/2$ to $1/2$ rather than from $0$ to $1$. Convergence time is reduced significantly with these easily implemented changes. Dahl (1987) improved the convergence rate using line search techniques. The learning rate is dynamically computed such that the minimization of the error function is a globally descending process. Smieja and Richards (1988) improve the learning performance by gradually increasing the difficulty of the problem domain. They change the learning parameters to develop a faster and more stable gradient descent by gradually deforming the shape of the error surface from a smooth to the final form. Samad (1988 and 1991) suggests modifying the generalized delta rule as follows

$$\Delta w_{ij} = \xi (o_i + \delta_j) \delta_j$$

whenever the $i$th unit is not an output unit. The idea behind this suggested variant of the learning rule is to use the expected output value for the $i$th unit which is $o_i + \delta_j$ instead of its current output $o_i$ to compute the weight increment $\Delta w_{ij}$.

Heuristic methods based on varying the learning rate during the training phase show good convergence rate (Cater 1987, Minai and Williams 1990, and Jacobs 1988). Jacobs (1988) introduced a method, called the delta-bar-delta algorithm for adjusting the learning rate to achieve faster convergence while adhering to the local computations. In this scheme, every weight has its own learning rate and these rates are varied based on the characteristics of the error surface. The modifying scheme for the learning rate $\xi_{ij}$ corresponding to the weight $w_{ij}$ is

$$\xi_{ij}^k = \xi_{ij}^{k-1} + \Delta \xi_{ij}^k$$

$$\Delta \xi_{ij}^k = \begin{cases} \alpha & \text{if } \frac{\partial E}{\partial w_{ij}} \frac{1}{\xi_{ij}} > 0 \\ -\omega \xi_{ij}^k & \text{if } \frac{\partial E}{\partial w_{ij}} \frac{1}{\xi_{ij}} < 0 \\ 0 & \text{otherwise} \end{cases}$$

where $\frac{\partial E}{\partial w_{ij}}$ is the partial derivative of the total error $E$ with respect to $w_{ij}$ computed at the $k$th learning iteration and $\xi_{ij}^k$ is basically an average of the current and past derivatives defined as follows

$$\bar{g}_{ij} = (1 - \beta) \frac{\partial E}{\partial w_{ij}} + \beta \bar{g}_{ij}^{k-1}$$

and parameters $\beta$, $\omega$, and $\alpha$ are predetermined. According to the delta-bar-delta algorithm, when the gradient for a weight has the same sign for many learning iterations, then the learning rate for that weight is incremented by a constant $\alpha$, because this behavior indicates that a minimum lies ahead. If the gradient changes signs
for several consecutive iterations, then the learning rate is decremented by a proportion \( \omega \) of its current value, since this indicates that a minimum is being jumped over. The delta-bar-delta rule increases learning rates linearly, but decreases them exponentially. The reason for increasing linearly is to prevent learning rates from becoming too large too rapidly, while for the decreasing exponentially is to always keep them positive and allow fast decrease. Minai and Williams (1990) incorporate momentum adjustment to Jacobs’ delta-bar-delta rule in an attempt to increase the rate of learning and to prevent wild jumps caused by large learning rules. Their algorithm is called the \textit{extended-delta-bar-delta} learning rule. Fahlman (1989) introduces an algorithm called \textit{quickprop} that is a bridge between second order methods and heuristic techniques. The two basic assumptions of quickprop are: (1) the total error is quadratic with respect to each individual weight and (2) the change of error surface gradient for each weight is not affected by all the other weights. Based on these heuristics, the quickprop learning rule takes the form

\[
\Delta w_{ij}^k = -\frac{\xi g_{ij}^k}{\varepsilon g_{ij}^k + \left[ \frac{g_{ij}^k}{g_{ij}^{k-1} - g_{ij}^k} \right] \Delta w_{ij}^{k-1}}
\]

(A.32)

where \( g_{ij}^k \) is defined in Eqn. (A.30). Actually, the quickprop learning rule can be considered a generalized delta rule Eqn. (A.27) with a dynamically computed momentum constant. The quickprop algorithm is very simple, needs only local computations, and effectively increases the convergence rate over the standard generalized delta rule. However, it can suffer from the flat areas of the error surface and unbounded weight increments.

In addition to the techniques mentioned above, attempts have been made to control the training process of backpropagation by adding new hidden units and connections (Ash 1989, Fahlman and Lebiere 1990, and Tenorio and Lee 1989), deleting unnecessary hidden units and connections (Kanin 1990 and Mozer and Smolensky 1989), using different error metrics (Solla, et al. 1988), selective presentation of learning samples (Ohnishi, et al. 1990 and Cho and Kim 1993), and modifying the slope of the sigmoid output function (Izui and Pentland 1990 and Rezgui and Tepedelenlioglu 1990).
BIBLIOGRAPHY

A


B


C


D


E


**F**


**G**


**H**


216

ICANN'94 (1994). *Proc. of the Fourth Annual International Conference on Artificial Neural Networks*. Sorrento Congress Center, near Naples, Italy.


J


K


L


M

N


NEURONET'93 (1993). *Proc. of the International Conference on Neural Networks and Neurocomputing*. Prague, Czech Republic.


O


P


U


V


W


WCNN’94 (1994). *Proc. of World Congress on Neural Networks*. San Diego, California.


Y


Z


