THE METHOD OF MANUFACTURED UNIVERSES FOR TESTING UNCERTAINTY QUANTIFICATION METHODS

A Thesis

by

HAYES FRANKLIN STRIPLING IV

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2010

Major Subject: Nuclear Engineering

THE METHOD OF MANUFACTURED UNIVERSES FOR TESTING UNCERTAINTY QUANTIFICATION METHODS

A Thesis

by

HAYES FRANKLIN STRIPLING IV

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Marvin L. Adams
Committee Members,	Bani K. Mallick
	Ryan G. McClarren
Head of Department,	Raymond J. Juzaitis

December 2010

Major Subject: Nuclear Engineering

ABSTRACT

The Method of Manufactured Universes for Testing Uncertainty Quantification Methods. (December 2010)
Hayes Franklin Stripling IV, B.S Nuclear Engineering, Texas A&M
Chair of Advisory Committee: Dr. Marvin L. Adams

The Method of Manufactured Universes is presented as a validation framework for uncertainty quantification (UQ) methodologies and as a tool for exploring the effects of statistical and modeling assumptions embedded in these methods. The framework calls for a manufactured reality from which "experimental" data are created (possibly with experimental error), an imperfect model (with uncertain inputs) from which simulation results are created (possibly with numerical error), the application of a system for quantifying uncertainties in model predictions, and an assessment of how accurately those uncertainties are quantified. The application presented for this research manufactures a particle-transport "universe," models it using diffusion theory with uncertain material parameters, and applies both Gaussian process and Bayesian MARS algorithms to make quantitative predictions about new "experiments" within the manufactured reality. To test further the responses of these UQ methods, we conduct exercises with "experimental" replicates, "measurement" error, and choices of physical inputs that reduce the accuracy of the diffusion model's approximation of our manufactured laws.

Our first application of MMU was rich in areas for exploration and highly informative. In the case of the Gaussian process code, we found that the fundamental statistical formulation was not appropriate for our functional data, but that the code allows a knowledgable user to vary parameters within this formulation to tailor its behavior for a specific problem. The Bayesian MARS formulation was a more natural

emulator given our manufactured laws, and we used the MMU framework to develop further a calibration method and to characterize the diffusion model discrepancy. Overall, we conclude that an MMU exercise with a properly designed universe (that is, one that is an adequate representation of some real-world problem) will provide the modeler with an added understanding of the interaction between a given UQ method and his/her more complex problem of interest. The modeler can then apply this added understanding and make more informed predictive statements.

This thesis is dedicated to the unwavering love, encouragement, and support of my mother and father, Tammy and Hayes Stripling, my brother Ross Stripling, and my fiancé and future wife Jenny Baker.

ACKNOWLEDGMENTS

I would like to extend my warm appreciation to Dr. Adams, Dr. McClarren, and Dr. Mallick for their dedication and unwavering guidance and support through the course of this research. I am looking forward to continuing our work and to making further contributions to science and engineering. I am also grateful for the nuclear engineering community at Texas A&M University, with whom I have thoroughly enjoyed the last six years. Also, a special acknowledgment to the Department of Energy Computational Science Graduate Fellowship program (DOE CSGF - grant number DE-FG02-97ER25308), which provides strong support to its fellows and their professional development.

NOMENCLATURE

BMARS	Bayesian Multivariate Adaptive Regression Splines
GP	Gaussian process
GPMSA	Gaussian Process Model for Simulation Analysis
LANL	Los Alamos National Laboratory
MMU	Method of Manufactured Universes
PS&E	Predictive science and engineering
QOIs	Quantities of interest
UQ	Uncertainty quantification
V&V	Verification and validation

TABLE OF CONTENTS

Page

ABSTRACT iii

DEDICATION v

ACKNOWLEDGMENTS vi

NOMENCLATURE vii

TABLE OF CONTENTS viii

LIST OF FIGURES x

LIST OF TABLES xiv

1 INTRODUCTION 1

1.1 Introduction of the Method of Manufactured Universes 2
1.1.1 The MMU Framework 3
1.1.2 Designing the "Universe" Given a Real Problem of Interest . . 4
1.1.3 Example: A Particle-Transport Universe 5
1.2 Description of the Computational Models in the Particle-Transport Universe 7
1.2.1 The S_N Method in 1D Slab Geometry 7
1.2.2 The Diffusion Method in 1D Slab Geometry 8
1.3 Description of the UQ Methodologies 8
1.3.1 The GPMSA Algorithm 8
1.3.2 The BMARS Algorithm 12

2 ANALYSIS OF THE GPMSA SOFTWARE IN OUR SIMPLE UNIVERSE 17

2.1 Specification of the Inputs 17
2.2 Initial Results from the GPMSA Software 19
2.2.1 Calibration of the Uncertain Inputs 20
2.2.2 Calibration of the Uncertain Inputs in the Presence of Noisy Observation Data 22
2.2.3 Investigation of the Emulator 28
2.2.4 Generating Experimental Predictions 32
2.2.5 The Addition of a Hidden Variable to the Simple Universe . . 35

Page

2.3 Further Investigation of the Model Discrepancy Term 40
2.3.1 Case 1: Calibrated Cross-sections are Contained Within the Input Space 41
2.3.2 Case 2: Simulator Calibration Requires Uncertain Inputs Outside the Given Range 45
2.4 Some Conclusions After Simple Tests of the GPMSA Software 49

3 ANALYSIS OF THE BMARS ALGORITHM APPLIED TO THE SIMPLE UNIVERSE 51

3.1 Choosing the Proper Settings for the BMARS Emulator 51
3.2 Calibrating the Emulator to the Experimental Data 52
3.3 Generating Predictions of New Experiments Using the BMARS Emulator 58
3.4 Demonstration of the Filtering Method in the Presence of Experimental Replicates 61
3.5 Analysis of the Discrepancy Function Using Ensembles of BMARS Models 66
3.6 Some Conclusions After Development of the BMARS/Weighing/Filtering Method 77

4 CONCLUSIONS AND DISCUSSION 79

REFERENCES 82

VITA 83

LIST OF FIGURES

FIGURE Page

1.1 The MARS algorithm requires exactly three splines to interpolate piecewise-linear training data. 14

1.2 A classical MARS fit to noisy training data cannot provide information about that noise. 15

1.3 A Bayesian MARS fit to noisy training data provides predictive distributions. 16

2.1 "Experimental" measurements and diffusion calculations of the reflected particle flow rate as a function of slab width. 18

2.2 For the case of "perfect measurements," GPMSA produced highly refined posterior distributions of the uncertain inputs to the diffusion model. . . 21

2.3 Calibrated diffusion calculation compared to "Experimental Measurements." 22

2.4 Input simulation results and noisy "experimental" observations. 23

2.5 The posterior distribution of the uncertain inputs in the case of noisy observed data has a notably larger spread than in the case of perfect measurements. 24

2.6 Calibrated diffusion calculation with noisy "experimental measurements." 26

2.7 Characterization of the observation noise hyperparameter. 27

2.8 The emulator is accurate at training data, x=[1,2,4,8,16,32]cm, but interpolates poorly at testing data, x=[3,6,10,12,14,18,22,26]cm. 29

2.9 The addition of experimental or simulation training points will help shape the prediction curve. 30

2.10 The model discrepancy term in our two test cases. 32

FIGURE Page

2.11 Experimental predictions are not as accurate at the training data when observation noise is included. 34

2.12 We add experimental replicates to the training data by performing 5 "measurements" of the reflected flow rate at each slab width. The difference in the replicates is the angular distribution of the incident flux. . . . 36

2.13 Predictions of the reflected particle flow rate as a function of slab width for default hyperparamter values. 37

2.14 The hyperparameter λ_y, which estimates the precision of experimental data, was "encouraged" towards a posterior distribution with smaller magnitude to reflect uncertainty introduced by the experimental replicates. The default prior distribution is $\Gamma(1, 0.001)$; our modified prior was $\Gamma(1, 1000)$. 39

2.15 The result of smaller values of λ_y is a wider predictive distribution of experimental results. 40

2.16 Predictions of the reflected particle flow rate as a function of slab width in the case of the scattering ratio, c=0.20. 42

2.17 The posterior distribution of the uncertain inputs in the case of a decreased scattering ratio has shifted far from the mean of the priors. . . . 43

2.18 True and estimated model discrepancy. We know that a model discrepancy does exist if the simulation is run at the "true" $\vec{\theta}$'s; the GPMSA algorithm, however, prefers to sample away from these "true" values and maintain a near-zero model discrepancy. 44

2.19 Experimental and simulation input data for the case of constricted range of the uncertain parameters. 45

2.20 The posterior distribution of the uncertain inputs in the case of a decreased scattering ratio and restricted uncertain input space. 46

2.21 True and predicted model discrepancy term. 47

2.22 Experimental predictions in the case dominated by the model discrepancy term. 48

FIGURE Page

3.1 The BMARS fit to the simulation data is most accurate at **I**=1, o=1. . . 53

3.2 Samples of the uncertain inputs are calibrated to the experimental measurements. 54

3.3 Likelihood distribution of the uncertain inputs at each experimental slab width (red/blue colors correspond to larger/smaller likelihoods). 56

3.4 The bivariate combined probability density function for the sampled $\vec{\theta}$ coordinates. 58

3.5 Assigning weights to the input space improves confidence intervals and slightly increases predictive accuracy. 60

3.6 Likelihood distribution of the uncertain inputs at each experimental slab width with multiple experimental replicates (red/blue colors correspond to larger/smaller likelihoods). 62

3.7 The global bivariate weight distribution in the case of 5 experimental replicates. 63

3.8 Calibrated diffusion simulator predictions in the case of multiple replicates. 64

3.9 Calibrated diffusion emulator predictions in the case of multiple replicates. 65

3.10 Input data for a scattering ratio of c=0.20. 67

3.11 Predictions made from the calibrated emulator and simulator will require a model discrepancy function to replicate the experimental results. . . . 68

3.12 The calibrated uncertain inputs in the case of c=0.20 over all slab widths. 69

3.13 Location and results from using different percentiles of the weighted inputs. 71

3.14 Training data for the BMARS discrepancy model. 73

3.15 Predictions of the discrepancy function for a range of BMARS sample sizes. 74

3.16 Experimental predictions in the case of a model discrepancy. 75

3.17 Decreased predictive variance resulting from a more refined sample of uncertain inputs. 76

LIST OF TABLES

TABLE Page

1.1 Description and default prior distributions for GPMSA hyperparamters . 10

2.1 Specification of experimental and simulation input parameters in the first simple "universe" 17

1. INTRODUCTION

The past decade has seen rapid advancement in complex computational projects and increasing dependence on these projects to support high-consequence decisions. An immediate result of this trend is the need for improved uncertainty quantification (UQ) methods to accompany the scientific simulations such that they not only deliever the best estimate of some quantity of interest, but also a measure of uncertainty in that estimate. In this research we are interested in UQ methods and software that attempt to predict experimental outcomes based on: 1) previous experimental data, and 2) simulations that use imperfect models that contain uncertain parameters. Recently developed methods, such as Bayesian inference and spectral representation, have been proposed for and applied to UQ in scientific computing by inferring functional relationships between simulation results and experimental measurements and then extending these relationships to forecast uncertainty in a new simulation or experiment [1] [2] [3].

Accurate simulation of an existing experiment is a challenge in itself; predictions of new experimental setups can be considerably more difficult, especially given little and/or uncertain experimental data. Nonetheless, increasingly complex systems coupled with advances in computer and software design will drive the need for predictive computational science and engineering. These predictions may take the form of a probabilistic statement, such as "With 95% confidence, we predict that this parameter will not come within 10% of its limit at these operating conditions." If important decisions are to be based on such statements, then all methods, models, and software that inform such statements – including the UQ methods, models, and software – should first be subjected to rigorous verification and validation assessments. These assessments provide the basis for statements of this type and should enable an unbiased knowledgeable evaluator to determine their credibility.

This thesis follows the style of Journal of Computational Physics.

With this motivation in mind, we present the Method of Manufactured Universes (MMU) as a framework that facilitates a comprehensive validation study of a given UQ method, perhaps as implemented in a given software system. To apply MMU, one defines the laws that govern a system, uses the laws to construct "experimental" results, simulates the "experiments" (either existing or new) using some computational model, and then tests the ability of the given UQ method to quantify the difference between simulation and "reality". This thesis presents results from a computationally simple yet rich "universe" in which two UQ methodologies are examined: first, a Gaussian process code from Los Alamos National Laboratory (LANL) and second, a Bayesian Multivariate Adaptive Regression Spline (BMARS) technique combined with a filtering/weighting method. The conclusion drawn from these results is that MMU is a powerful technique that can help identify problems in UQ software, help computational scientists and engineers understand the subtleties, strengths, and weaknesses of various UQ methodologies, and help decision-makers evaluate the credibility of predictive statements.

1.1 Introduction of the Method of Manufactured Universes

The motivation for this framework is the need to understand the assumptions embedded in UQ methods and the manner in which the effects of these assumptions propagate to the method's output. For example, a common practice for describing an unknown distribution (prior distribution or output uncertainty, for example) is to assume a Gaussian distribution with some estimated mean and standard deviation. The underlying function, however, may have only finite support and may be asymmetric; this information could be lost, excluded, or misrepresented by the assumed normal distribution. In some applications, this may be an acceptable approximation. It can easily be imagined, however, that a certain problem or set of physics might not be accurately represented in this manner.

We emphasize that it is not the purpose of MMU to expose every flaw or invalid assumption of every UQ method. Assumptions and limitations in statistical processes are often well known and documented. Instead, our purpose is to propose and illustrate a framework that others may use to determine the applicability of a given method to their specific problems. We champion the idea of "glass box" approaches to uncertainty quantification, and we believe that the simple study presented in this paper is an excellent example of the value added in understanding the mechanics of the predictive software.

1.1.1 The MMU Framework

The following list presents the basic steps of the MMU framework.

1. Define laws that govern the manufactured universe. This means creating mathematical models that define the laws that govern system behavior and the physical constants that serve as inputs to the models. As discussed below, these laws should reflect some key characteristics of modeler's real problem of interest.

2. Create "experiments" by defining physical problems and use the manufactured laws to create exact output quantities of interest (QOIs). Then create "measured" data by perturbing these output QOIs using an error model.

3. Define an approximate model on which the UQ methodology is to be tested. This will include the choices of input parameters to the simulation and estimates of their uncertainties (these estimates, themselves, could be uncertain)

4. Apply the given UQ methodology to the set of {approximate model, uncertain input constants, measured data}.

5. Define a new set of experiments and predict new values of the QOIs, with uncertainties, using what was learned from the UQ methodology. Generate "real" experimental results using the manufactured laws, and assess how well the UQ method quantified the uncertainties in the predictions.

Of course, this method can be repeated with variations on the approximate models, measurement-error models, data uncertainties, UQ methodology parameters, and universal laws.

1.1.2 Designing the "Universe" Given a Real Problem of Interest

To maximize the utility of an MMU study, the modeler should try to manufacture a universe (that is, the physical laws, "experiments," approximate model, and the relationship between them) that properly reflects the physics, computational models, and uncertainties of the real-world predictive science or engineering problem. Further, the universe must also allow the modeler to explore, isolate, and further understand the characteristics of the UQ methodology. We identify two important factors that the modeler should consider when manufacturing the universe:

1. The manufactured universe should contain the same *types* and *sources* of uncertainty as the real-world problem.

   Unless care is taken with this, there is a risk that the lessons learned from using MMU may not apply to the real problem of interest. Some characteristics of the real problem that one might seek to mimic in the manufactured universe

include the mix of epistemic and aleatory uncertainties, the nature of prior distributions of uncertain parameters, and the origin and path of propagation of the most important uncertainties.

2. The simplifications that lead from the manufactured laws to the MMU approximate model should be closely related to and simplifications of the real-world physics that lead to the real-world mathematical model of interest.

   As we emphasized before and will show by example, a complete treatment of uncertainty – which includes uncertainty due to model error – must be informed by the physics of the problem. Therefore, the model error in the MMU analysis should closely mimic the model error in the real problem to maximize the real-world value of the insight gained through the MMU analysis.

1.1.3 Example: A Particle-Transport Universe

The examples in this paper will present results from a neutral particle transport "universe." The universe is relevant to our real-world problems in which transport calculations play key roles in the analysis of complex systems such as nuclear reactors or high energy-density laboratory experiments. Uncertainties in these real problems are often driven by material properties (such as interaction cross-sections), and we are often interested in the model fidelity and accuracy of material properties that are required to produce quantities of interest such as material temperatures and particle fluxes within a desired confidence interval. To address these questions, we shall include cross-sections as our uncertain parameters, employ a higher and lower fidelity calculation as our "reality" and approximate model, respectively, and investigate two UQ methodologies in which we are interested for our real-world applications.

Our manufactured reality is that our particle-transport universe behaves exactly according to the equations of the S_8 discrete ordinates method in one-dimensional slab geometry with no spatial discretization (see section 1.2.1). We select specific values for the following physical constants for each "experiment" in our "universe": total cross-section, Σ_t, scattering ratio, $c = \Sigma_s/\Sigma_t$, incident angular flux, ψ_{inc}, and slab thickness, x. Using the S_8 method, we conduct "experiments" in which we "measure" the reflected and transmitted particle flow rates (denoted Y_1 and Y_2 respectively) resulting from the specified angular incident flux on one side of the slab.

Next we define analytic diffusion theory as our approximate model (see section 1.2.2). The uncertain inputs to diffusion, denoted $\vec{\theta}$, are the diffusion coefficient, $\theta_1 = D = 1/(3\Sigma_t)$, and the absorption cross-section, $\theta_2 = \Sigma_a$. The components of $\vec{\theta}$ are defined on a prior probability density function, $\pi(\vec{\theta})$. In our case, this prior is uniform. The response data, $Y_{1,sim}$, is a function of the slab thickness and uncertain inputs, $Y_{1,sim} = f(x, \vec{\theta})$, where f is defined by the diffusion solution.

As mentioned in the introduction, the UQ models studied in this paper are the Gaussian Process Model for Simulation Analysis (GPMSA) code from LANL and a BMARS implementation coupled with a calibration procedure developed at Texas A&M. These methodologies both attempt to serve two major functions. First, they attempt to produce a calibrated posterior distribution of $\vec{\theta}$ that favors the combinations of the uncertain inputs that are most likely to accurately predict experimental measurements. Second, they provide methods to extend this calibration and make predictions of new experimental measurements, including uncertainties.

To exercise these two methods, we will attempt to predict Y_1 at new slab thicknesses for which experimental data did not previously exist. We will then compare these predictions to "measurements" from the S_8 calculation at the new thicknesses.

1.2 Description of the Computational Models in the Particle-Transport Universe

The experimental data will be generated using analytic solutions of the S_8 equations; the simulation will be computed using analytic solutions of the diffusion equation. The next two sections outline these equations and their solutions in 1D slab geometry.

1.2.1 The S_N Method in 1D Slab Geometry

The manufactured experimental data used in this analysis is generated using the one-group discrete ordinates method. The governing equation in 1D slab geometry with no volumetric source is

$$\mu_m \frac{d\Psi_m(x)}{dx} + \Sigma_t \Psi_m(x) - \frac{c\Sigma_t}{2} \sum_{n=1}^{N} \Psi_n(x) w_n = 0$$

where Ψ_m is the angular flux in the m^{th} quadrature direction. In the S_8 method, we have 8 directions whose cosines are given by an 8-point Gauss-Legendre quadrature set on the interval (-1,1). The analytic solution is a sum of exponential terms [4] [5]:

$$\Psi_m(x) = \sum_{k=1}^{N} A_k \frac{\nu_k}{\nu_k - \mu_m} \exp\left(\frac{-\Sigma_t x}{\nu_k}\right)$$

where each of the N distinct ν_k's satisfies the relation

$$\frac{2}{c} = \sum_{m=1}^{N} w_m \frac{\nu}{\nu - \mu_m}.$$

The N constants, A_k, are obtained by requiring that the solution satisfy the boundary and interface conditions. In our universe, we define an incident angular flux at each boundary and require continuity of the Ψ_m's at region interfaces. The "experimental response" of the reflected particle flow rate is calculated as

$$Y_1 = j^-(x = 0) = \sum_{\substack{m \\ \mu_m < 0}} |\mu_m| \Psi_m(0) w_m.$$

In some iterations of the MMU framework, we will generate "measured data" by perturbing these calculations with an error model.

1.2.2 The Diffusion Method in 1D Slab Geometry

The analytic solution to the diffusion equation in a finite 1D slab is simple and well known. The governing diffusion equation with no volumetric source term is written as:

$$\frac{d^2\Phi(x)}{dx^2} - \frac{1}{L^2}\Phi(x) = 0 \quad 0 \leq x \leq L_s$$

where L is the diffusion length, $L = \sqrt{D/\Sigma_a}$, and D is the diffusion coefficient which we will define as $D = 1/(3\Sigma_t)$. We express the analytic solution in terms of the hyperbolic functions:

$$\Phi(x) = C_1 \text{sinh}\left(\frac{x}{L}\right) + C_2 \text{cosh}\left(\frac{x}{L}\right)$$

where constants C_1 and C_2 satisfy the known incident particle flow rates on each of the slab boundaries:

$$j^+(0) = \frac{\Phi(0)}{4} + \frac{J(0)}{2}$$
$$j^-(L_s) = \frac{\Phi(L_s)}{4} - \frac{J(L_s)}{2}$$

and $J(x)$ is the net particle flow rate, which in diffusion theory satisfies $J(x) = -D\frac{d\Phi}{dx}$. The reflected particle flow rate, $Y_{1,sim}$, can be calculated as

$$Y_{1,sim} = j^-(0) = \frac{\Phi(0)}{4} - \frac{J(0)}{2}.$$

1.3 Description of the UQ Methodologies

1.3.1 The GPMSA Algorithm

The GPMSA code from Los Alamos National Laboratory is a multivariate extension of the Kennedy and O'Hagen univariate (*i.e.*, scalar output) formulation for predictions of experimental measurements [1]

$$Y(\vec{x_i}) = \eta(\vec{x_i}, \vec{\theta}) + \delta(\vec{x_i}) + e_i, \tag{1.1}$$

where Y is the experimental value being computed or estimated, $\vec{x_i}$ is a vector of independent inputs, $\vec{\theta}$ is a vector of calibration parameters, $\eta(\vec{x_i}, \vec{\theta})$ is an emulator of the simulation response to inputs $\vec{x_i}$ and $\vec{\theta}$, $\delta(\vec{x_i})$ is a model-discrepancy function, and e_i is a random error term. If each term in Eq. (1.1) is a distribution rather than a single-valued function, then we obtain a an expected prediction $\langle Y \rangle$ and a distribution about $\langle Y \rangle$ that characterizes uncertainty in the prediction.

The GPMSA algorithm builds stationary Gaussian processes (GPs) for both the emulator and discrepancy term with covariance matrices of the following general form [3]:

$$\mathrm{cov}_\eta((\vec{x}, \vec{\theta}), (\vec{x}', \vec{\theta}')) = \frac{1}{\lambda_\eta} \exp\left\{ -\sum_{k=1}^{p} \beta_k^\eta |x_k - x_k'|^\alpha - \sum_{k'=1}^{l} \beta_{p+k'}^\eta |\theta_{k'} - \theta_{k'}'|^\alpha \right\}, \quad (1.2)$$

$$\mathrm{cov}_\delta(\vec{x}, \vec{x}') = \frac{1}{\lambda_\delta} \exp\left\{ -\sum_{k=1}^{p} \beta_k^\delta |x_k - x_k'|^{\alpha_\delta} \right\}. \quad (1.3)$$

The emulator covariance function, cov_η depends on both the independent and uncertain inputs and the discrepancy covariance, cov_δ, depends only on the independent inputs. In Eqs. (1.2) and (1.3), p is the number of independent inputs, l is the number of uncertain inputs, and α and α_δ are shaping parameters, which equal 2 to form the stationary Gaussian covariance kernel.

The hyperparameters λ_η, λ_δ, β^η, and β^δ shape the covariance function and determine the characteristics of the model's predictions. In general, the β parameters are scaling vectors that determine the strength of the covariance dependence in each input dimension, and the λ parameters are proportional to the inverse of the variance of the input data and therefore are a measure of the precision of the inputs. The GPMSA algorithm explores distributions of the hyperparameters, as described below; an informed modeler can encourage this exploration by tailoring prior distributions, $\pi_{\mathrm{pr}}(\cdot)$, of these parameters to his specific problem. Table 1.1 lists a selection of the GPMSA hyperparameters and their default prior distributions.

Table 1.1
Description and default prior distributions for GPMSA hyperparamters

Hyper-parameter	**Description**	**Prior Distribution Shape**	**Default Parameter Values**
β^{η}	Simulation Scaling Vector	Calculated as $(\rho_{\eta})^{a}$	N/A
β^{δ}	Discrepancy Function Scaling Vector	Calculated as $(\rho_{\delta})^{a}$	N/A
λ_{η}	Marginal Simulation Precision	$\Gamma(\lambda_{\eta}\vert$ a,b$)^{b}$	a=5, b=5
λ_{δ}	Marginal Discrepancy Precision	$\Gamma(\lambda_{\delta}\vert$ a,b$)^{b}$	a=1, b=10^{-5}
λ_{y}	Observed Data Noise Precision	$\Gamma(\lambda_{y}\vert$ a,b$)^{b}$	a=1, b=10^{-3}
λ_{ys}	Simulation Data Noise Precision	$\Gamma(\lambda_{ys}\vert$ a,b$)^{b}$	a=5, b=5*10^{-3}
ρ_{η}	β^{η} Transformation Variable	$\beta(\rho_{\eta}\vert$ α,b$)^{c}$	α=1(fixed), b=0.2
ρ_{δ}	β^{δ} Transformation Variable	$\beta(\rho_{\eta}\vert$ α,b$)^{c}$	α=1(fixed), b=0.2

[a] $\rho_i = \exp\left(\frac{-\beta^i}{4}\right)$
[b] $\Gamma(\lambda|a,b) \propto \lambda^{a-1}\exp(b\lambda)$
[c] $\beta(\rho|\alpha,\mathrm{b}) \propto \rho^{\alpha-1}(1-\rho)^{\mathrm{b}-1}$

The algorithm constructs a fully-Bayesian formulation to infer the posterior distribution, $\pi_{\mathrm{ps}}(\cdot)$ of $\vec{\theta}$ and the hyperparameters

$$\begin{aligned}&\pi_{\mathrm{ps}}(\vec{\theta},\lambda_{\eta},\lambda_{\delta},\beta^{\eta},\beta^{\delta}|\vec{z}) \propto \\ &\quad L\left(\vec{z}|\vec{\theta},\lambda_{\eta},\lambda_{\delta},\beta^{\eta},\beta^{\delta}\right)\pi_{\mathrm{pr}}\left(\vec{\theta}\right)\pi_{\mathrm{pr}}\left(\lambda_{\eta}\right)\pi_{\mathrm{pr}}\left(\lambda_{\delta}\right)\pi_{\mathrm{pr}}\left(\beta^{\eta}\right)\pi_{\mathrm{pr}}\left(\beta^{\delta}\right).\end{aligned} \tag{1.4}$$

The first n elements of $\vec{z}$ contain the experimental responses and the last m contain simulation responses. The likelihood structure is the standard multivariate normal density

$$L\left(\vec{z}|\vec{\theta},\lambda_\eta,\lambda_\delta,\beta^\eta,\beta^\delta\right) \propto |\Sigma_z|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}\vec{z}^{\,\mathrm{T}}\Sigma_z^{-1}\vec{z}\right). \tag{1.5}$$

The covariance matrix is defined as

$$\Sigma_z = \Sigma_\eta + \begin{pmatrix} \Sigma_y+\Sigma_\delta & 0 \\ 0 & 0 \end{pmatrix},$$

where Σ_η is formed by applying Eq. (1.2) to the $n+m$ experimental and simulation inputs (with candidate $\vec{\theta}$ values inserted for the n experimental inputs), Σ_δ is obtained by applying Eq. (1.3) to the n experimental inputs, and Σ_y is the $\mathbb{R}^n$ observation covariance matrix. The general model is developed in more thorough detail in Higdon *et. al.* (2004) [3] and its multivariate extension is developed in Higdon *et. al.* (2008) [2].

The distribution (1.4) is explored via a Metropolis [6] Markov chain Monte Carlo (MCMC) algorithm with Hastings updates [7]. In simple terms, the algorithm goes as follows (where we will call $\vec{\Omega}$ the vector of uncertain inputs and hyperparameters):

1. Initialize $\vec{\Omega}^1$

2. Given current iterate, $\vec{\Omega}^k$, sample candidate $\vec{\Omega}^*$ such that $\mathbb{P}(\vec{\Omega}^k \rightarrow \vec{\Omega}^*)$=$\mathbb{P}(\vec{\Omega}^* \rightarrow \vec{\Omega}^k)$, where $\mathbb{P}(x^1 \rightarrow x^2)$ is the probability of selecting x^2 given the current iterate x^1.

3. Compute the Metropolis acceptance probability

$$\alpha = \min\left\{1, \quad \frac{\pi_{\mathrm{ps}}(\vec{\Omega}^*|\vec{z})}{\pi_{\mathrm{ps}}(\vec{\Omega}^k|\vec{z})}\right\} \tag{1.6}$$

4. Choose $\vec{\Omega}^{k+1}$:

$$\vec{\Omega}^{k+1} = \begin{cases} \vec{\Omega}^{*} & \text{with probability } \alpha \\ \vec{\Omega}^{k} & \text{with probability } (1-\alpha) \end{cases}$$

The acceptance criteria, Eq. (1.6), is designed such that candidates with larger likelihoods are always accepted (α=1); candidates with smaller likelihoods may also be accepted, but with a smaller (although never non-zero) probability. We are interested in accepting less-likely candidates in the interest of thorough mixing; that is, we avoid being 'stuck' in local maxima of the posterior distribution and have a non-zero probability of jumping far (in some sense) away in the domain to explore other regions of the posterior.

Realizations from the MCMC process are used to construct the distributions in Eq. (1.1) using standard multivariate normal theory. The process also provides estimates of calibrated distributions of the uncertain inputs, $\vec{\theta}$, such that the simulator is more likely to replicate experimental results when run with draws from these distributions.

1.3.2 The BMARS Algorithm

The Multivariate Adaptive Regression Splines (MARS) algorithm is a partition-based curve-fitting technique which attempts to emulate the mapping between a function's inputs and outputs as a summation of spline functions. Multivariate spline functions are simply products of one-dimensional spline functions; these 1D spline functions are defined to be zero on part of the domain and a polynomial of some order on the remainder of the domain. The knot of the spline is the coordinate at which this definition changes, and the direction of the spline describes whether the non-zero portion of the spline is in the positive or negative direction from the knot. In our implementation, knot points are restricted to the coordinates of the training data given to the model.

Given a set of input or training data, the classical formulation [8] uses a semi-stochastic method to generate a basis function of the form:

$$B(x) = \beta_0 + \sum_{k=1}^{\mathbf{K}} \beta_k \prod_{l=0}^{\mathbf{I}} (x_l - t_{k,l})_+^{o_k} \tag{1.7}$$

where $\vec{x}$ is a vector of inputs, $t_{k,l}$ is the knot point in the l^{th} dimension of the k^{th} component, the function $(y)_+$ evaluates to y if $y > 0$, else it is 0, o is the polynomial degree of the k^{th} component, β_k is the coefficient of the k^{th} component, $\mathbf{K}$ is the maximum number of components of the basis function, and $\mathbf{I}$ is the maximum allowed number of interactions between the L dimensions of the input space. Note that the formulation does not require that each of the k components have a term in each dimension of $\vec{x}$. Also, the function has only a maximum of $\mathbf{K}$ components, but implementations typically search for the optimal fit while minimizing k.

As an example of a MARS fit, consider Fig. 1.1. Imagine that we have a 1D simulation which returns piecewise-linear data (shown as the blue scatter-points in the figure). A MARS fit of the form (1.7) would require a summation of at least three splines to produce an exact fit: the constant spline β_0, and two other 1D splines, at least one of which must have a knot coordinate where the training data changes slope.

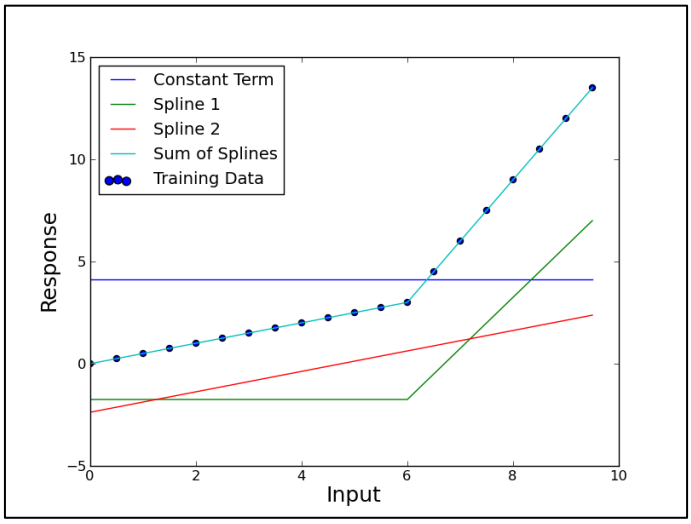


Fig. 1.1. The MARS algorithm requires exactly three splines to interpolate piecewise-linear training data.

The Bayesian extension of MARS (hereby named BMARS) [9] uses a Markovian process to propose new components of the basis function (new splines), delete existing splines, or modify existing splines. When a new spline is created, the algorithm randomly chooses its order, knot point, direction, and level of interaction. The Markov chain algorithm iterates this random selection process and accepts/rejects proposal basis functions based on a likelihood calculation. This likelihood is a function of the candidate's fit to the training data and the number of splines in its basis function. The coefficients, β, are found via matrix inversion, but are perturbed on a noise distribution that is updated as the process evolves. Similar to the GPMSA software, a number of samples from the converged Markov chain can be used to generate a predictive distribution of the underlying function's response at new inputs.

To illustrate the utility of BMARS, consider an extension of the simple 1D regression above to the case where the simulation produces noisy response data, as shown in Fig. 1.2. A classical MARS fit, depending on the user-defined tuning parameters,

could be made to generate a line of best fit (as shown) or to interpolate every training point. If we wanted to use this regression to predict the next simulation output, however, neither of these results would be very helpful. The line of best fit would predict an idea of the expected simulation response, while the complete interpolation gives no credit to the noise at all!

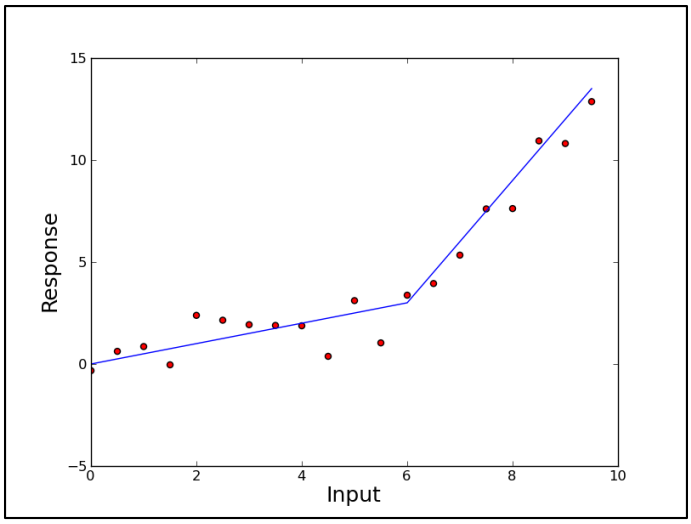


Fig. 1.2. A classical MARS fit to noisy training data cannot provide information about that noise.

Given data of this type (which is often the norm in our complex problems), we'd prefer to compute a predictive distribution of the simulation response at a new input. This would provide an estimate of the expected value as well as a confidence interval about that value. Figure 1.3 illustrates a BMARS fit to the same noisy training data, where a number of splines from the posterior distribution of MARS splines has been plotted to illustrate the predictive distributions. We see that some splines tend to chase the noise while others trace the mean or expected value, and the end result is a distribution at any point where the emulator is evaluated. These kinds of

distributions are necessary for characterizing uncertainty in both the simulation and the regression of the simulation.

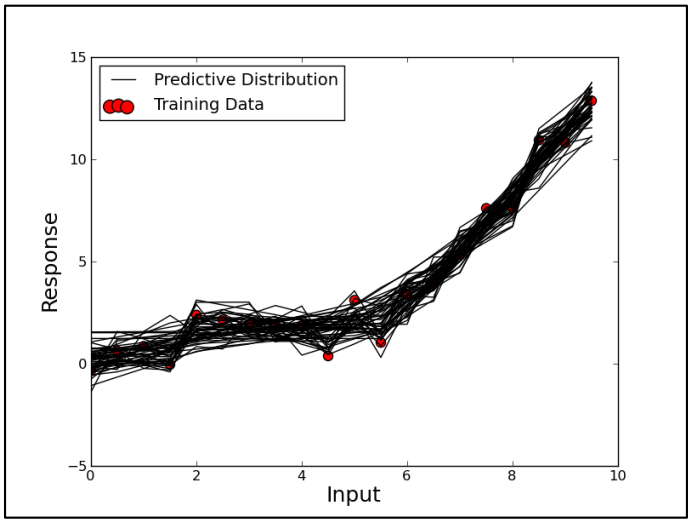


Fig. 1.3. A Bayesian MARS fit to noisy training data provides predictive distributions.

In the application to our universe, we construct a BMARS basis function to emulate the diffusion simulator. We then attempt to calibrate the simulation to the "experimental measurements" by tuning the uncertain cross-section inputs. We first randomly sample the $\vec{\theta}$ input space and use the BMARS emulator to construct a distribution of predicted simulation responses at each experimental slab width. We assign a weight to each sampled $\vec{\theta}$ coordinate based on the density of its predictive distribution evaluated at the "experimental measurement." Then, through interpretation of the weights at each slab width or subset of slab widths, we can choose the simulation inputs that are most likely to produce accurate predictions of a "measurement" for some new "experiment" and provide a measure of uncertainty in that prediction.

2. ANALYSIS OF THE GPMSA SOFTWARE IN OUR SIMPLE UNIVERSE

For our first exercise of the MMU framework, we generate data in the simplest setting available: experimental data with no imposed error or noise and no hidden variables. We select the GPMSA software as the first UQ model.

2.1 Specification of the Inputs

Table 2.1 summarizes the input parameters for the mathematical models.

Table 2.1
Specification of experimental and simulation input parameters in the first simple "universe"

"Experimental" Inputs	
Slab Widths, x	[1 2 4 8 16 32]cm
Σ_t, c	1.00 cm^{-1}, 0.99
Incident Flux, Ψ_{inc}	$\Psi_m(0) = 1.0, \quad \mu_m > 0$ $\Psi_m(L_s) = 0.0, \quad \mu_m < 0$
Simulation Inputs	
Slab Widths, x	[1 2 4 8 16 32]cm
$\theta_1 = \text{D}$	$\theta_1 \in$[0.133 0.533]cm
$\theta_2 = \Sigma_a$	$\theta_2 \in$[0.004 0.016]cm^{-1}
θ Sample Space	Full tensor product, each θ sampled at 5 evenly distributed points on domain, including endpoints.
Incident Particle Flow Rate	$j^+(0) = \sum_{\substack{m \\ \mu_m>0}} \Psi_m(0)\mu_m w_m$ $j^-(L_s) = \sum_{\substack{m \\ \mu_m<0}} \Psi_m(L_s)\lvert\mu_m\rvert w_m$

The result of this input space is six experimental results and 150 simulation results for the response function, Y_1. The six experimental results are perfect "measurements" of the experimental response at each slab width; the 150 simulation results come from 25 sampled $\vec{\theta}$ points at each of the six slab widths. The incident angular flux is isotropic from the left side of the slab and zero from the right side. Figure 2.1 shows the relationship between the results of the manufactured "experiments" and simulations for each input slab width.

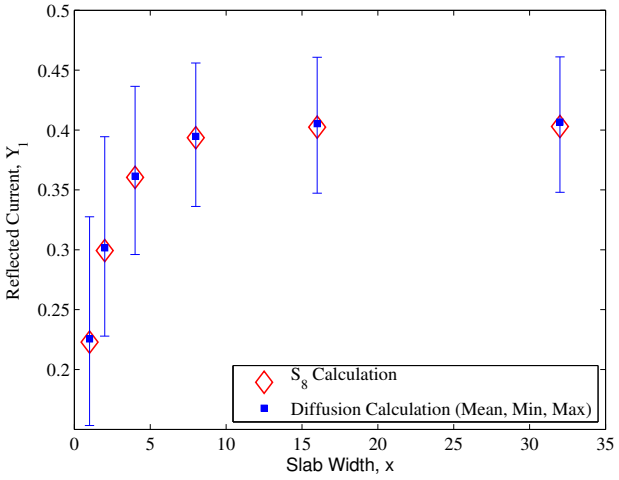


Fig. 2.1. "Experimental" measurements and diffusion calculations of the reflected particle flow rate as a function of slab width.

It is important to recognize that (in the case of high scattering ratio) the mean of the diffusion simulations accurately approximate the "experimental measurements" for each slab width. Thus, in the context of the Kennedy/O'Hagan formulation (Eq. (1.1)), we will not expect a large model-discrepancy contribution, $\delta(\vec{x})$. This simplifies the calibration portion of the problem. Also, we intentionally chose the

range of the components of $\vec{\theta}$ such that the simulation results spread comfortably around the "experimental" results.

2.2 Initial Results from the GPMSA Software

Initially the input data is formatted and given to the GPMSA software in a black-box fashion: we did not take advantage of knowledge of the embedded statistical models and made no attempts to modify the hyperparameter priors. In the analysis that follows, we have chosen to focus on three major objectives of the GPMSA software:

1. Calibration of the uncertain inputs to their most likely distribution (that is, the values of $\vec{\theta}$ that are most likely to result in accurate predictions of experimental outcomes);
2. The ability of the emulator, $\eta(\vec{x_i}, \vec{\theta})$, to accurately characterize the simulation;
3. The ability of the software to predict the response of a new experiment, $Y(\vec{x_i})$, within a reasonable confidence interval.

At this point we review the naming conventions used in the following analysis:

- *training* data - Experimental settings/measurements and simulation inputs/outputs provided to the software
- *testing* data - Experimental settings/measurements and simulation inputs/outputs that were not provided to the software and might be predicted
- *prior* distribution - The probability density function initially assumed by the software in the MCMC algorithm
- *posterior* distribution - The likelihood distribution produced via the MCMC process.

2.2.1 Calibration of the Uncertain Inputs

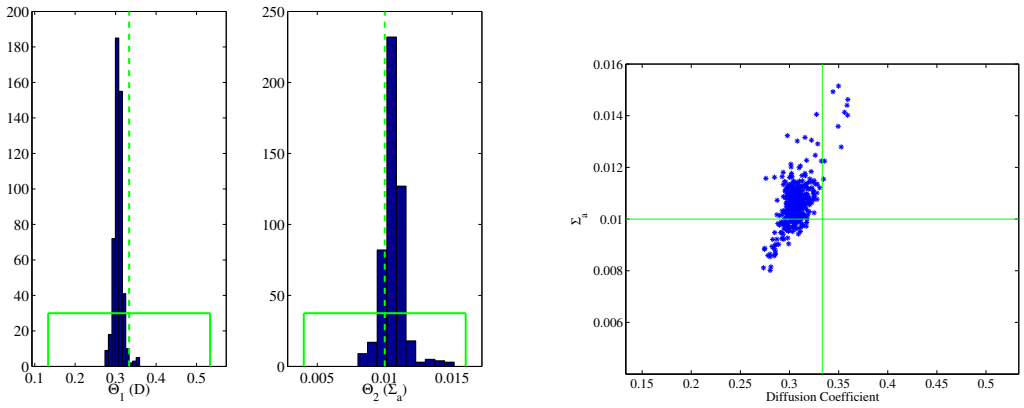
A major function of the code (and, in fact, the focus of the work by Kennedy and O'Hagan [1]) is to calibrate $\vec{\theta}$ such that the simulation will more accurately approximate experimental measurements. In their original work, Kennedy and O'Hagan specified the $\vec{\theta}$ parameters as tuning parameters, such as coefficients on error terms or empirical data correlations which did not necessarily have a physical meaning. In our application, however, the uncertain inputs correspond to physical properties of the material. Thus, we seek to infer a range of these physical values that result in diffusion simulations that accurately predict observed "experimental" results and not necessarily those that accurately describe the material in the physical problem.

The distribution of $\vec{\theta}$, Eq. (1.4), is explored via the Markov chain process. A sample of the MCMC realizations (after convergence) can be drawn and run through the simulator to determine whether the posterior distribution of $\vec{\theta}$ more closely predicts experimental response at the training data. In the analysis that follows, the Markov chain was run to 10,000 iterations with a 7,000 iteration burn-in. A uniform sample of 500 realizations was taken from the last 3,000 iterations to generate the results.

By its default settings, the GPMSA software assumes a normal prior distribution on each of the the uncertain inputs. This distribution has a mean equal to the mean of the training sample of the uncertain input and a standard deviation equal to ten times the range of the uncertain input. This formulation results in (nearly) equal weighting of the full range of the input but allows (when possible) for a refinement in the uncertain input space to a tighter Gaussian posterior distribution. Figure 2.2(a) illustrates the posterior likelihood distribution (shown as the nearly-Gaussian histogram) suggested for the uncertain inputs, D and Σ_a. Immediately, we note that the range of the calibrated uncertain inputs has been reduced considerably from the original range of the input values (shown as the uniform distributions). This suggests

that the Markov chain has focused on a distribution of the parameters that it believes will more accurately replicate experimental results.

Figure 2.2(b) gives a two-dimensional illustration of the inferred distribution of the uncertain inputs. An interesting observation is that the posterior distribution of the diffusion coefficient is centered about a value below the mean value (which is the usual diffusion-theory value that corresponds to the transport reality, namely $1/3\Sigma_t$) of the prior. This is an indication by the algorithm that, for this specific problem, the simulation (diffusion) is more likely to match experimental data when the diffusion coefficient is biased below the usual diffusion-theory value. The posterior distribution for Σ_a, however, seems to be centered at or near the mean of the prior, indicating that the simulation does not prefer a biased value for this θ parameter.



(a) Uniform prior and nearly-Gaussian posterior distribution of the uncertain inputs

(b) Location and concentration of the calibrated uncertain inputs

Fig. 2.2. For the case of "perfect measurements," GPMSA produced highly refined posterior distributions of the uncertain inputs to the diffusion model.

Figure 2.3 is a reconstruction of Figure 2.1 except that the simulation response data is calculated based on samples of the posterior distribution of $\vec{\theta}$. We immediately

notice that the range of computed particle flow rate at the training slab widths is reduced in the calibrated simulation. In this sense, the algorithm has succeeded in calibrating the inputs to reduce the uncertainty of its predictions.

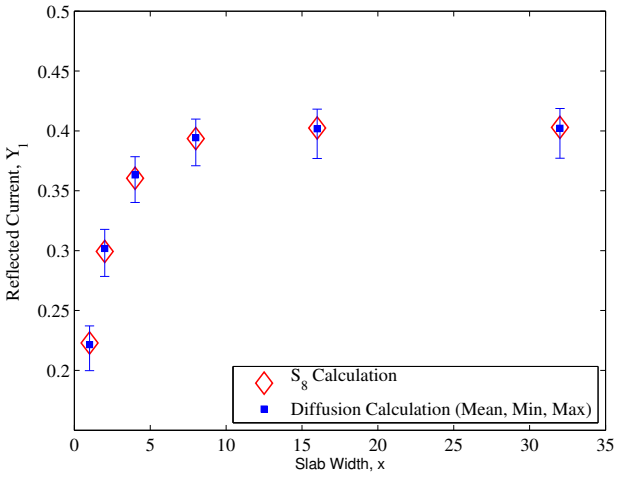


Fig. 2.3. Calibrated diffusion calculation compared to "Experimental Measurements."

2.2.2 Calibration of the Uncertain Inputs in the Presence of Noisy Observation Data

This initial illustration of the GPMSA software has assumed that the experimentalist measures the observed data without any error or bias. In practical applications, however, the norm is incomplete or noisy data sets resulting from a limited number of available experiments. The presence of noise in observed data can result from a number of factors, including epistemic uncertainty (unknown unknowns), aleatory uncertainty (natural stochastic behavior), bias introduced by the measuring equipment itself, or bias introduced by the experimental setup or procedure. No matter

the cause of noisy observation data, the result is increased difficulty (and therefore added uncertainty) in generating experimental predictions.

In many cases, especially those in which a large number of experiments and measurements are documented, the nature of the observation error can be well characterized by an error function such as that in Eq. (1.1). In other cases, we might expect the Bayesian nature of the GPMSA algorithm to recognize noise in the data and infer a different distribution of hyperparameters for the Gaussian process. To simulate measurement error, the same MMU exercise as described in the previous section was repeated with the addition of a normally distributed error term to the observed data, $e \sim \mathrm{N}(0,0.05^2)$. The resulting input data is illustrated in Fig. 2.4 (compare to Fig. 2.1):

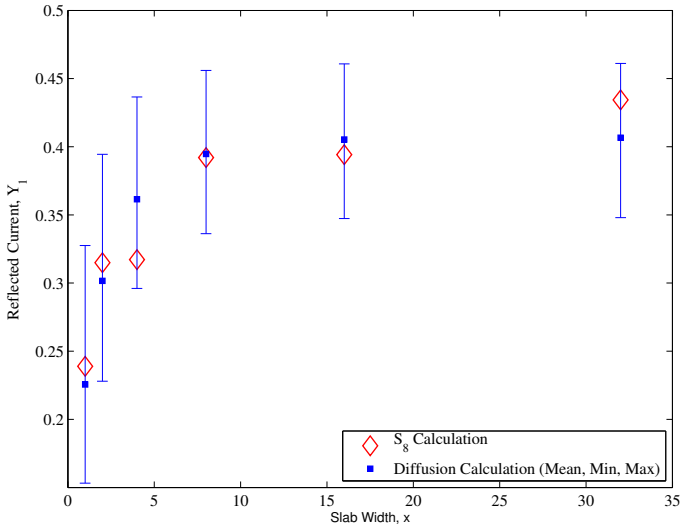
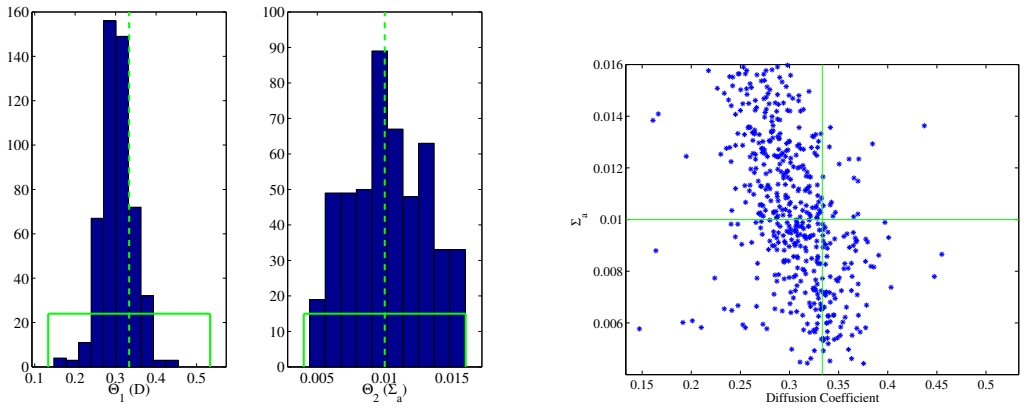


Fig. 2.4. Input simulation results and noisy "experimental" observations.

Note that the diffusion simulation input data and response are exactly the same as in the previous exercise. We note that the magnitude of the observation error,

although fairly large and perhaps unrealistic for such a simple problem, is meant to magnify the response of the GPMSA algorithm to the presence of noisy data. The calibrated vector $\vec{\theta}$ in this case is illustrated in Figs. 2.5(a) and 2.5(b).



(a) Posterior distribution of the uncertain inputs in the case of noisy experimental data

(b) Location and concentration of the calibrated uncertain inputs in the case of noisy experimental data

Fig. 2.5. The posterior distribution of the uncertain inputs in the case of noisy observed data has a notably larger spread than in the case of perfect measurements.

When compared to Figs. 2.2(a) and 2.2(b), Figs 2.5(a) and 2.5(b) illustrate a clear reaction from the calibration software to the noisy observed data. The posterior distribution of the uncertain inputs have a noticeably larger standard deviation, especially in the case of θ_2, the absorption cross-section. The GPMSA software does not allow the MCMC algorithm to sample the uncertain inputs outside the domain of their input range. In the case of θ_2, it seems clear that the MCMC chain would continue to expand the distribution if the bounds were widened. It is also interesting to note that the distribution of the diffusion coefficient is again centered below the mean value of the input data (accept with much wider variance).

Figure 2.5(b) indicates that the calibrated simulator in the case of this noisy experimental data is sensitive to the value of θ_1 and that the preferential value of θ_1 may have a weak dependence on the value of θ_2 (indicated by the slight trend in the scatter). Aside from this weak trend, the calibrated simulator does not appear to have preferential values of the absorption cross-section, θ_2. Finally, we see evidence that the algorithm is "chasing" some of the noise in the observed data because of the larger number of outlying scatter-points in Fig. 2.5(b) compared to Fig. 2.2(b).

Figure 2.6 (compare to Fig. 2.3) shows the calibrated diffusion results at the training data. The range of simulation results is considerably larger than the calibrated simulator with no observation noise and only slightly smaller than the uncalibrated simulator. Also, at the slab width of 4cm, the range of simulation results does not encompass the noisy observation. At this point, it is unclear whether the algorithm will treat this point as an outlier or attempt to use the discrepancy formulation to 'chase' this noise. This will be explored in a later section.

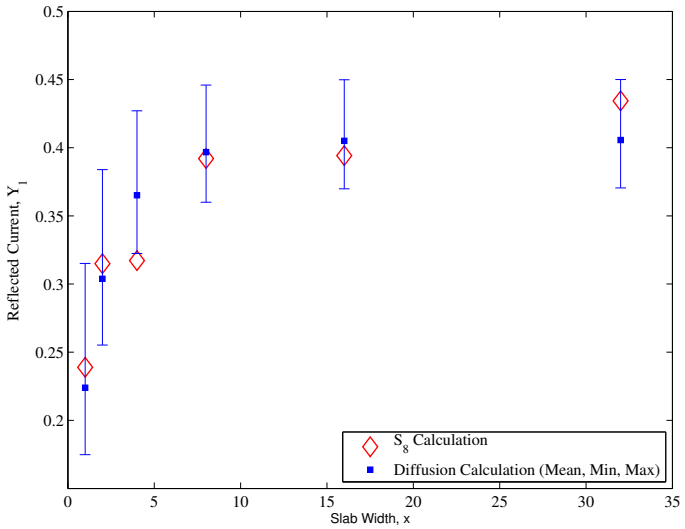


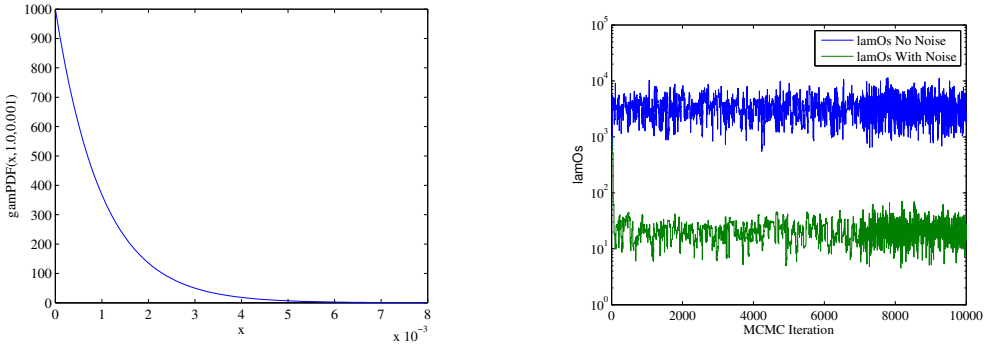
Fig. 2.6. Calibrated diffusion calculation with noisy "experimental measurements."

As mentioned previously, the construction of the fully Bayesian Gaussian process has more than one method for accounting for noise in the observed data. The widened distributions shown in Fig. 2.5 will result in wider response predictions by the emulator in effort to account for the observation noise. A statistical treatment of the observation noise occurs in the construction of the covariance kernel described by Eq. (1.5). The parameter λ_y (referred to as *lamOs* in the GPMSA documentation [10]) is a measure of the precision of the observation data, which is inversely proportional to the variance of the observation data. In this fully Bayesian implementation, we would expect the algorithm to find smaller magnitudes of this parameter more likely in the case of noisy observation data.

This is the first example of the potential benefits gained through an understanding of the UQ code. In a 'black box' implementation of this problem, the user would either assume that the algorithm will recognize the noise in the observation error and treat it properly or, worst case, would not even consider the the issue at all.

Alternatively, for the user with a working familiarity with the method, the GPMSA software allows for adjustments to the prior distributions on the hyperparameters, including λ_y. Thus, if the distribution of the observation error is known *a priori* (which is typically the case), the user can adjust this prior distribution and thereby add one more tie between the UQ method and the physics.

In the interest of pursuing the MMU framework, we do not alter the default prior of the observation noise hyperparameter at this point. In other words, the case with and without noise added to the experimental data were run with the same priors on the noise hyperparameter to test whether or not the GPMSA software would be able to differentiate between the cases. The default prior is a gamma distribution, $\lambda_y \sim \Gamma(1.0,0.001)$, and is illustrated in Fig. 2.7(a).



(a) The default prior for the observation noise hyperparameter - a gamma distribution.

(b) The algorithm responds to noisy experimental data by accepting lower values of the noisy hyperparamter.

Fig. 2.7. Characterization of the observation noise hyperparameter.

Figure 2.7(b) illustrates the accepted value of *lamOs* as a function of Markov chain iteration (note the log scale). The algorithm correctly prefers larger values of the hyperparameter in the case of no noise in the experimental measurements and smaller values of the hyperparameter in the case with noise. This is an encouraging

result and, as evidenced by Fig. 2.6 and discussed in a later section, will result in wider experimental prediction distributions at testing inputs.

2.2.3 Investigation of the Emulator

The emulator, $\eta(\vec{x_i}, \vec{\theta})$, is a regression algorithm designed to predict simulation results at testing data points. In many applications of the Kennedy and O'Hagan model, an accurate emulator can be an advantage because the scientific simulation is computationally expensive. In our simple universe, an emulator is hardly necessary from a computational cost-saving perspective. We are interested, however, in how accurately the UQ methodology's emulator represents the simulation model. Because our simulations predict experimental results (*i.e.* measurement error ignored) with reasonable accuracy (see Figure 2.1), we expect that an accurate emulator is the key component to accurate predictions of experimental outcomes at testing data points and therefore proceed to evaluate its performance.

Figure 2.8 illustrates the calculation of the reflected particle flow rate by both the emulator and the simulator as a function of slab thickness. The simulation data are the diffusion response as a function of slab width using the 500 $\vec{\theta}$ draws (i.e., calibrated $\vec{\theta}$ draws) used to plot the posterior distributions in Fig. 2.2(a) . The emulator data are computed via the methods embedded in the GPMSA software using hyperparameters realizations from the same 500 MCMC draws.

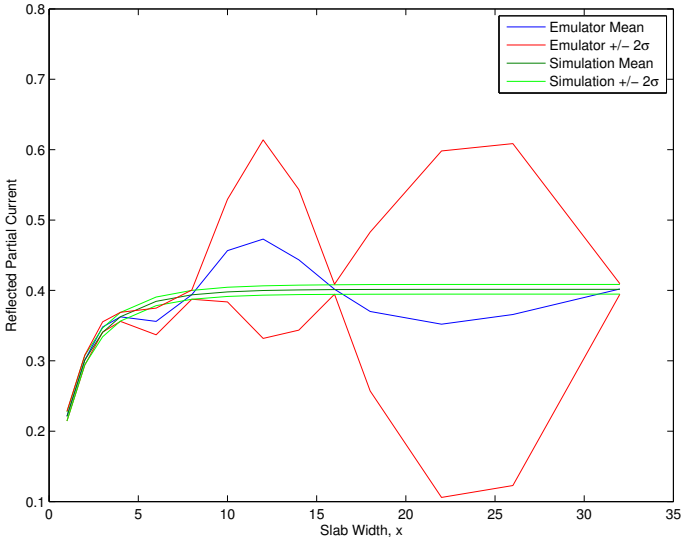


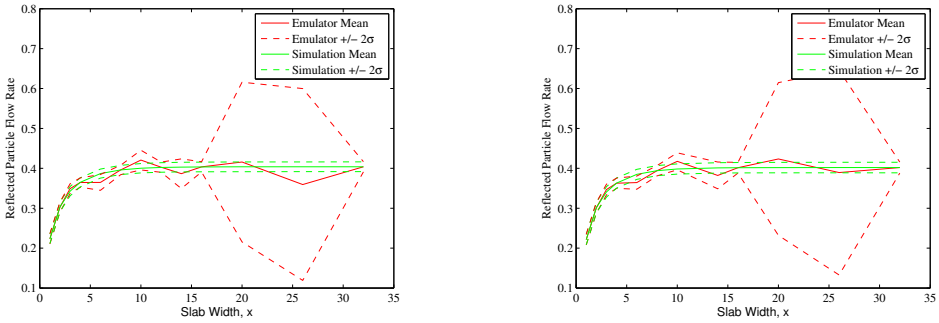
Fig. 2.8. The emulator is accurate at training data, x=[1,2,4,8,16,32]cm, but interpolates poorly at testing data, x=[3,6,10,12,14,18,22,26]cm.

Several features of this plot are worth mentioning. First, at the training data, both the mean and standard deviation of the simulation response are predicted accurately. Second, relating to the previous sections, the simulation runs using the calibrated (posterior) distributions of $\vec{\theta}$ are smooth and have small variance, indicating that the calibration procedure has decreased the simulation output variance (we will assess the predictive accuracy of the simulator in a following section). Third, at testing data points, the standard deviation of the emulator predictions does increase, as expected, but perhaps more dramatically than one might expect for such a well-behaved system. The mean of the emulator predictions, however, wanders far from the simulator values at the testing data points, especially those that are far away from training points.

This is a behavior commonly observed in static Gaussian process models, which is the type of GP employed by this algorithm. The model is termed static because the value of the scaling and variance hyperparamters which construct the covariance

model do not depend on the location of the prediction point in the input domain. A consequence of this construction is the inability to distinguish between smooth and rapidly-varying portions of the response surface. To make this distinction, the scaling hyperparameter would have to account for the decreasing correlation between input points as the function moves from smooth to rapidly varying domains. The MCMC procedure is designed to "learn" the optimized distribution of the hyperparameters given available samples from the simulation, but the algorithm cannot account for non-static behavior in the response function.

An obvious (but not always feasible) technique to improve the regression is to add data points to the training set. For example, in a scenario in which we are especially interested in experimental predictions at a slab width of 12cm, we may be able to afford one additional experiment or simulation at that value. Figure 2.9 illustrates the predictive effect of adding data points to the training set (namely, a simulation or experiment at x=12cm).



(a) Adding an Experimental Training Point at x=12cm

(b) Adding a Simulation Training Point at x=12cm

Fig. 2.9. The addition of experimental or simulation training points will help shape the prediction curve.

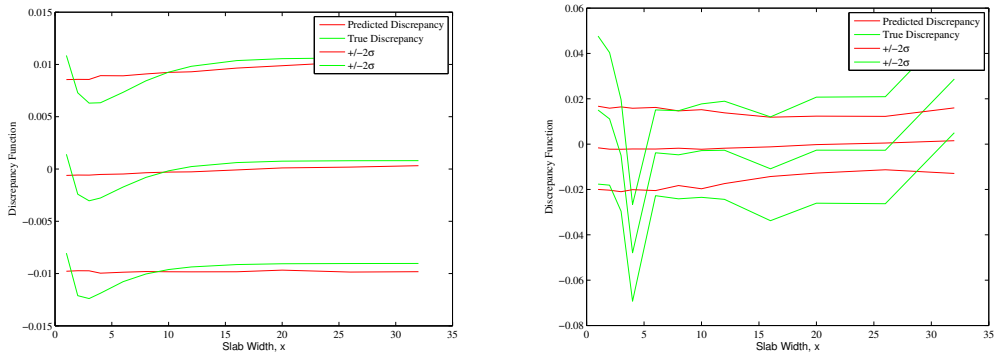
The figure indicates that, in this problem, the behavior of the mean of the prediction curve behaves similarly after an addition to the experimental or simulation training data set. In the context of this problem, this is explained by the fact that magnitude of the model discrepancy function, $\delta(\vec{x})$, is small compared to the magnitude of the response function. Therefore, experimental and simulation training data are similar and have similar effects on the correlation matrix. In a problem framework involving a systematic or random model discrepancy, we would expect additions to the experimental training data set to have a more profound effect on the prediction curve than additions to the simulation training data set.

Another feature of Fig. 2.9 is that the standard deviation of the prediction at $x = 12$cm is smaller when that data point is added to the experimental training data set than when added to the simulation data training set. In some sense, this indicates that the software is more confident in predictions made at existing experimental training input data than in existing simulation training input data. We find this behavior intuitive and were pleased at this result.

Finally, and most importantly, we note that in both cases, predictions near $x = 12$cm did improve, but the behavior of the prediction curve is still incoherent at the remaining testing data points. Although the magnitude of the oscillations did decrease, the predictions made by the software are still outside a reasonable range of acceptability. In other words, we were able to improve predictive capability around the data point of interest, but overall the GP method is still unable to resolve the difference between the rapidly-varying and saturated domain of the response function. A number of modifications or extensions to the GP method have been developed, including the formulation of non-stationary covariance models [11] and methods such as "blind kriging" [12]; future work would certainly include the application of the MMU framework using the same kinds of predictive validation problems with these UQ methods.

2.2.4 Generating Experimental Predictions

We now complete the Kennedy/O'Hagan formulation by adding the model discrepancy term and generating full experimental predictions. In the case of high scattering ratio, the model discrepancy term is very small in magnitude compared to the emulator. Figures 2.10(a) and 2.10(b) illustrate the "true" and predicted model discrepancy term in the case of perfect and noisy observation measurements, respectively. Here we exercise our MMU framework and compute the "true" model discrepancy as simply the mean simulation response subtracted from the "observed" experimental response. Note that calculations at the set of testing slab widths, x_{tst}=[3,6,10,12,20,26], is included in the figures.



(a) Model discrepancy term in the case of perfect experimental measurements.

(b) Model discrepancy term in the case of noisy observation data.

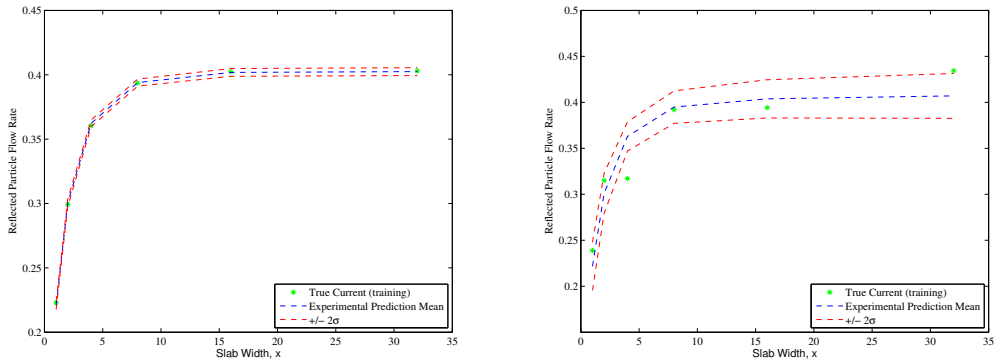
Fig. 2.10. The model discrepancy term in our two test cases.

These plots have a number of interesting features. First, in the case with no noise in the experimental data, the "true" and predicted model discrepancy terms agree very closely accept at slab widths below 10cm. It is well known that the diffusion approximation to transport breaks down near boundary layers, so we might expect

some model error at optically thin slab widths. Except at the slab width of 1cm, Fig. 2.10(a) indicates that the diffusion approximation is over-predicting the S_8 calculation. This is not readily observable in Fig. 2.1 because the magnitude of the discrepancy is very small compared to the magnitude of the response function.

Figure 2.10(b) is a bit more interesting. We see the oscillations in the "true" model discrepancy that result from the noise in the observation data, the largest of which is again at x=4cm slab width. The predicted model discrepancy, however, does not chase this noise, indicating that the algorithm has identified noise in the observation data and will not chase it. The predicted curve stays very near zero throughout the domain of the problem which indicates that the experimental predictions will essentially be equal to the emulator results.

Figures 2.11(a) and 2.11(b) illustrate the experimental predictions *at the training data* made by the GPMSA software for the case with and without experimental measurement error. It is important to note that the GPMSA predictions *do not* include the noise term e_i given in Eq. (1.1); that is, GPMSA attempts to predict the true physical response, not the measurement made by the observer. If we wished to predict the experimental measurement, we would simply add the distribution of e_i, which we claim is typically well characterized, to the GPMSA predictions.



(a) Experimental predictions at the training data with no observation noise.

(b) Experimental predictions at the training data with added observation noise.

Fig. 2.11. Experimental predictions are not as accurate at the training data when observation noise is included.

Again we highlight that the shape of the experimental predictions is essentially the same as the emulator results because the model discrepancy term is very near zero in these test cases. In the case with no noise on the observation data, we see that the experimental predictions are *exactly* correct. Indeed this is a property of a well-defined Gaussian process regression on a well-behaved function: the regression function exactly fits the training data. This exact fit is not the case with the noisy observation data where the algorithm has not chased the noise but instead has tried to smooth out the shape.

We also now see the effect of the decreased magnitude of the observation noise precision hyperparameter λ_y and widened posterior distribution of the uncertain inputs on the magnitude of the uncertainty bounds about the experimental predictions. The response is intuitive - the uncertainty added to the system by the observation noise results in more uncertainty in the experimental predictions. The code seems to have resisted chasing the noise in the observation data, but it is likely that tuning of the hyperparameters could result in more or less chasing of this noise.

2.2.5 The Addition of a Hidden Variable to the Simple Universe

Usually, theoretical models do not fully encompass the physics of the true experimental phenomenon. This may be a result of simplifying assumptions in the model, simplifications in the interest of computational feasibility, or a lack of understanding of the science under investigation. In such cases, the ability to quantify the added uncertainty resulting from the model inadequacy becomes extremely important.

In the interest of investigating model inadequacy, we will impose a hidden variable on our particle transport universe. Instead of a single experimental result at each slab width, we will replicate each experiment using five different distributions of the incident angular flux. The S_8 method will "measure" different reflected particle flow rates for each incident distribution. Our computational model, diffusion theory, will not know the difference between the replicates, as we will specify the incident flux distribution such that diffusion theory always has the same boundary conditions (i.e., each replicate has the same incident particle flow rate but different angular distribution). Therefore, to the computational model, the differences in experimental replicates will look like experiment-to-experiment variability, while in reality, there is a hidden physical mechanism causing the different "measurements" at each slab width.

Figure 2.12 shows the experimental and simulation input data (compare to Fig. 2.1). Again, we have chosen a wide enough prior distribution for the uncertain inputs that for a given slab width, the range of simulation results encompassed each of the five experimental replicates.

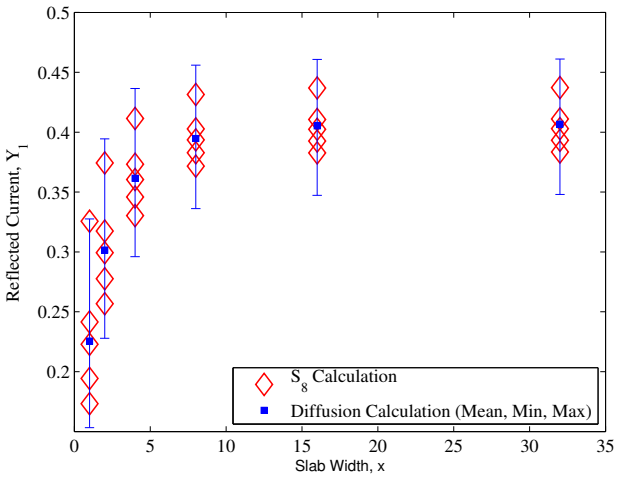


Fig. 2.12. We add experimental replicates to the training data by performing 5 "measurements" of the reflected flow rate at each slab width. The difference in the replicates is the angular distribution of the incident flux.

Figure 2.13 illustrates the experimental response predictions (no e_i term) made by the GPMSA software given this input data. Again, because in this "universe" there is little model discrepancy, the emulations and full predictions are essentially identical.

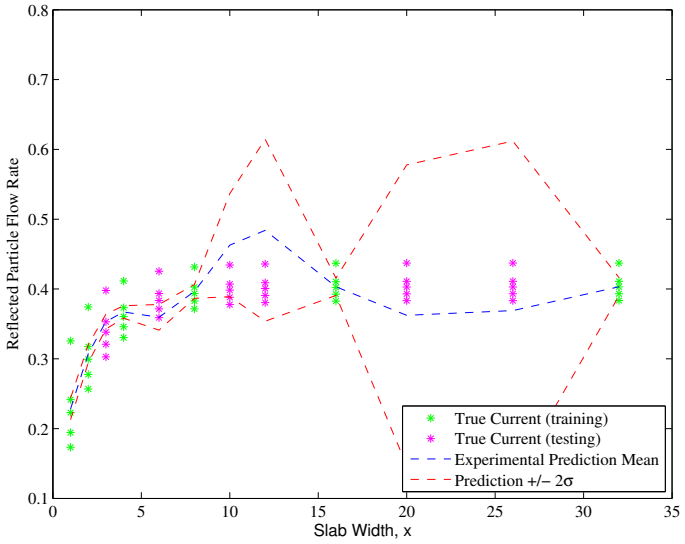


Fig. 2.13. Predictions of the reflected particle flow rate as a function of slab width for default hyperparamter values.

Again, several features of the prediction function are worth noting. First, as in Sec. 2.2.3, we see accurate predictions of the mean response at each training data point but poor interpolations and unrealistic oscillations at interpolating testing points. Second, we do not see an increase in the standard deviation of the response predictions as a result of the experimental replicates. In fact, at most training points, the plot indicates that three to four of the five replicates are outside two standard deviations of the experimental response predictions. Of course, the addition of an e_i term might result in the inclusion of these outlying results, but in our "universe", the experiments were measured perfectly! A hidden variable is responsible for the experiment-to-experiment variability, not measurement error or bias.

An accurate treatment of the experiment-to-experiment variability (in terms of the Kennedy and O'Hagan model) would lead to an increased variance in the model discrepancy term. Ideally, the software would recognize that the true model discrepancy has multiple values at each slab width and would reflect this uncertainty

by widening the distribution in $\delta(\vec{x})$. We do not see this behavior, however. Instead, it seems that the algorithm interprets the experimental replicates as noise or measurement error.

We believe this is a valuable example of the utility of MMU: If a modeler discovers that a certain source of uncertainty (in our case, differences in experiments to which our model is insensitive) is not being treated correctly by the UQ method, then he/she would have to make appropriate changes to the model or its inputs to better characterize this uncertainty. The GPMSA code provides methods for the knowledgeable user to make such changes, as we illustrate with the following example.

One of the Bayesian hyperparameters, *lamOs*, is a measure of the precision of the experimental data, where precision is proportional to the inverse of the variance. In our universe, we "measure" our results perfectly (*i.e.* no noise), implicating a relatively large value of *lamOs*. Larger values of the precision hyperparamters will tend to result in smaller predictive variances. We have seen, however, that the added variance caused by our experimental replicates is not being properly accounted for in the response predictions.

The GPMSA code allows the user to "encourage" the hyperparameters towards particular values by changing the prior distribution from which the parameters are sampled. In an attempt to widen the range of experimental predictions, we set the gamma prior distribution of *lamOS* towards smaller values (in essence, indicating we have a large variance in our experimental data). Figure 2.14 illustrates the Markov chain of *lamOs* before and after the change in prior and Fig. 2.15 shows the result of this change on the experimental predictions:

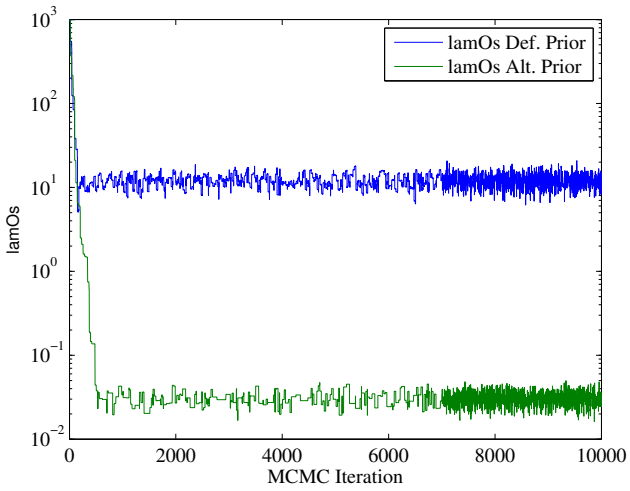


Fig. 2.14. The hyperparameter λ_y, which estimates the precision of experimental data, was "encouraged" towards a posterior distribution with smaller magnitude to reflect uncertainty introduced by the experimental replicates. The default prior distribution is $\Gamma(1, 0.001)$; our modified prior was $\Gamma(1, 1000)$.

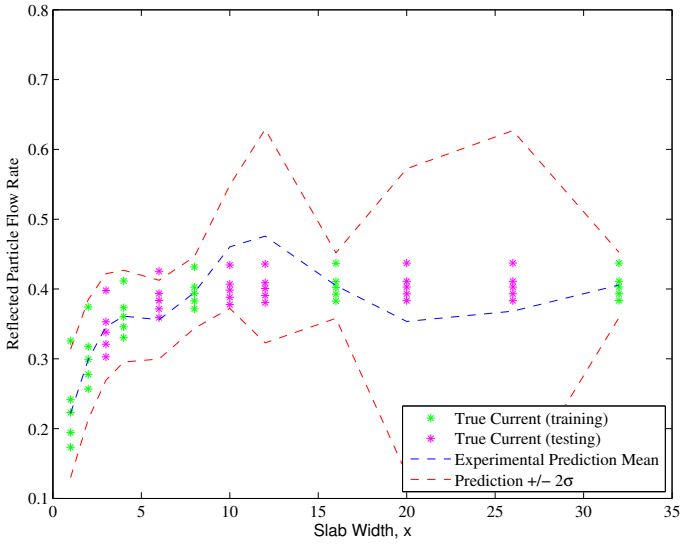


Fig. 2.15. The result of smaller values of λ_y is a wider predictive distribution of experimental results.

We did succeed in widening the 2σ confidence interval by altering the prior of the *lamOs* hyperparameter, as it now encloses most of the "experimental measurements." We were unable, however, to determine a set of priors on the other hyperparameters that would improve the shape of the prediction curve. For example, several attempts to modify the β_η scaling vector did result in changes to the testing data predictions, but none resulted in consistent accuracy across the domain.

2.3 Further Investigation of the Model Discrepancy Term

As the modeler in the MMU method, we have established the "exact" physics, constants, and inputs and can therefore know exactly the model discrepancy term. In actual scenarios, having this knowledge is highly unrealistic (indeed, if we knew exactly the model discrepancy, then we would know exactly the predictions or quantities of interest!). The model discrepancy term is often complicated with assumptions of

the model, unknown unknowns, inexact physics, and/or true randomness in nature. Thus it is equally difficult to characterize as the simulator and is often formulated with fewer training points than is the emulator.

One objective of this thesis is to subject the simple universe to scenarios involving a model discrepancy function and gain an understanding of how well the GPMSA software characterizes the term. This will be accomplished by decreasing the scattering ratio of the system which decreases the accuracy of the diffusion approximation to the transport calculation. For these tests we will remove the hidden variable from the system. The scattering ratio will be reduced to $c = 0.20$ and, to maintain a functional shape of the response, the training slab widths will be x_{trn}=[0.5,0.8,1.2,1.6,2.2,3.0]. Otherwise the input data is identical to that given in Table 2.1.

2.3.1 Case 1: Calibrated Cross-sections are Contained Within the Input Space

Figure 2.16 gives the input data for the first test case. Here we see a large discrepancy between the mean simulation result and the "experimental" results. Also, the mean simulation result is negative which is an unphysical quantity in the context of this problem.

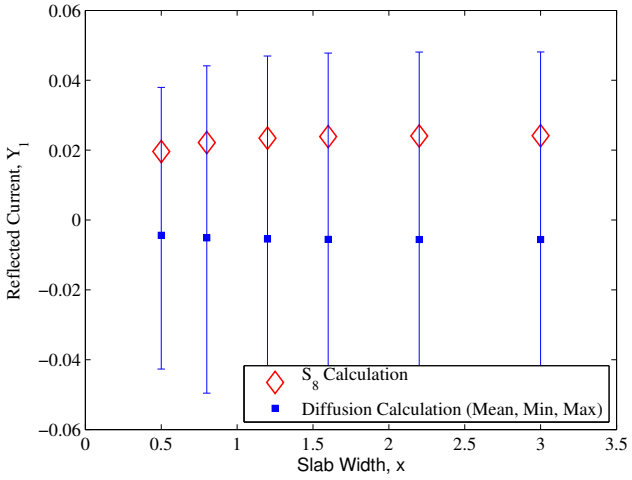
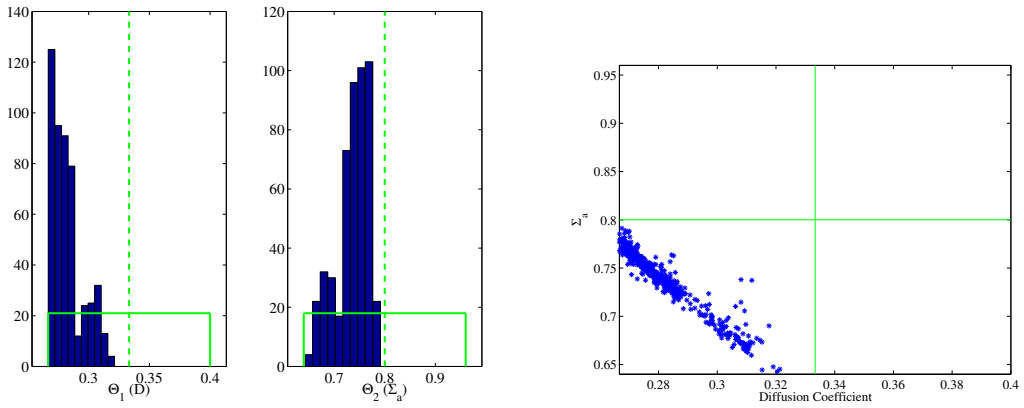


Fig. 2.16. Predictions of the reflected particle flow rate as a function of slab width in the case of the scattering ratio, c=0.20.

Figures 2.17(a) and 2.17(b) show the results of the calibration effort of the GPMSA software. We see a large change resulting from the decreased scattering ratio.



(a) Posterior distribution of the uncertain inputs in the case a decreased scattering ratio

(b) Location and concentration of the calibrated uncertain inputs in the case of a decreased scattering ratio

Fig. 2.17. The posterior distribution of the uncertain inputs in the case of a decreased scattering ratio has shifted far from the mean of the priors.

First, we note that the distribution is centered below the mean value of the uncertain input space, indicating that the diffusion simulator is more likely to replicate experimental measurements (in this case of $c = 0.20$) when the absorption cross-section and diffusion coefficient are biased below their true or typically-defined value, respectively. Also the association between the two uncertain inputs seems well-defined and sharp given the shape of the scatter in Fig. 2.17(b). Finally, as was the case with Σ_a in the "universe" with noisy experimental data, we cannot conclude the optimal value of $\vec{\theta}$ from the posterior distributions because the distributions have appreciable probabilities at the edges of the domain; in other words, we cannot conclude that the peak of the posterior distribution is contained by the bounds of the prior distribution.

The question of interest, however, is how the software treated the discrepancy function. We find that because the Markov chain has identified a region in the uncertain input space where simulation results closely match experimental results,

the software prefers to sample from this space and leave the discrepancy function very small. Figure 2.18 illustrates the nearly-zero model discrepancy term for this case:

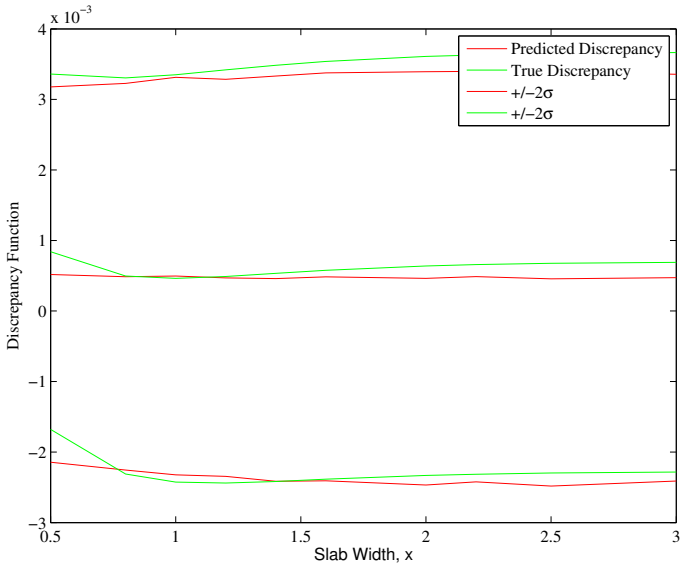


Fig. 2.18. True and estimated model discrepancy. We know that a model discrepancy does exist if the simulation is run at the "true" $\vec{\theta}$'s; the GPMSA algorithm, however, prefers to sample away from these "true" values and maintain a near-zero model discrepancy.

In the figure, the estimated model discrepancy function is calculated as the difference between the software's predicted experimental and simulation result at each slab width. The "true" discrepancy is calculated by subtracting the *actual* diffusion result at the input from the actual S_8 result at the same input. We see that the function is slightly positive and constant, but again it is small in magnitude compared to the response function (although a larger contributor than in the case of $c = 0.99$).

The conclusion of this test is that the software preferred to rely on an accurate calibration of the uncertain inputs (in this case, away from physically "true" values)

and emulator-dominant predictions rather than attempting to account for the model discrepancy. In the case where the uncertain inputs were non-physical tuning parameters, this exercise would highlight the calibrated value of these inputs that, in some sense, offset the model discrepancy.

2.3.2 Case 2: Simulator Calibration Requires Uncertain Inputs Outside the Given Range

Given the findings in the previous section, the obvious test is to require the GPMSA software to formulate a model discrepancy term by reducing the allowable range of the uncertain inputs. For this test, the uncertain input space will be reduced to $D \in [0.3267, 0.340]$ and $\Sigma_a \in [0.784, 0.816]$. The input data for run of the GPMSA software is shown in Fig. 2.19.

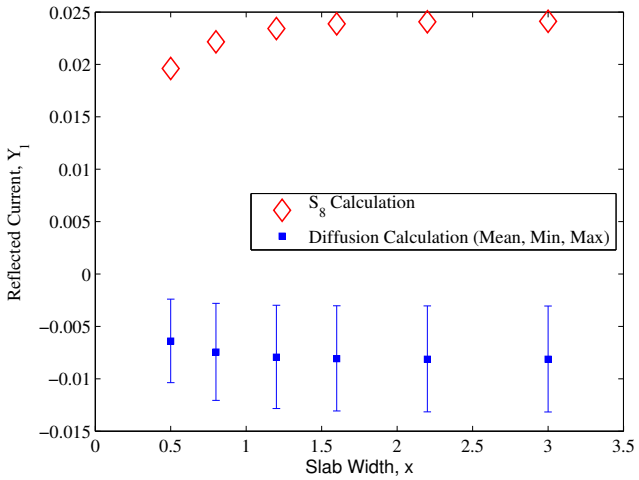
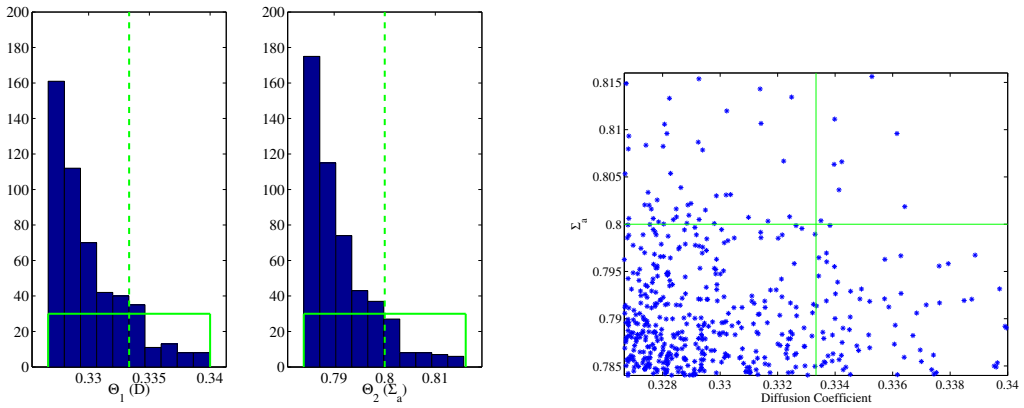


Fig. 2.19. Experimental and simulation input data for the case of constricted range of the uncertain parameters.

Again, we note that the mean results of the simulation are negative and unphysical. Also, compared to the simulation data in the previous section, the range of the simulation results is smaller, reflecting the smaller bounds on the uncertain inputs. The plot indicates that a model discrepancy function will be required to replicate the experimental data because the range of simulation results does not encompass the experimental data.

As presented in previous sections, the calibrated uncertain inputs are given in Figs. 2.20(a) and 2.20(b).



(a) Posterior distribution of the restricted uncertain inputs in the case a decreased scattering ratio and restricted range.

(b) Location and concentration of the restricted uncertain inputs in the case of a decreased scattering ratio

Fig. 2.20. The posterior distribution of the uncertain inputs in the case of a decreased scattering ratio and restricted uncertain input space.

Figure 2.20(b) resembles the behavior in Fig. 2.17(b) as the bulk of the concentration of the calibrated uncertain inputs is located in the lower-left corner of the plot. However, in this case where the range of the uncertain inputs is much smaller, the calibrated set is much noisier. This is a result of the fact that *none* of the uncertain inputs are much more likely than the others because none replicate

experimental results. In other words, we see some concentration below the means of the uncertain inputs because these values do produce simulation results that are closer to the experimental data, but they are not *much* more likely than the rest of the uncertain input space and therefore we see more acceptance of other candidates.

Because we manufactured the universe in which this data was generated, we know exactly the model discrepancy and can compare this knowledge to predictions from the GPMSA software. Figure 2.21 illustrates this comparison.

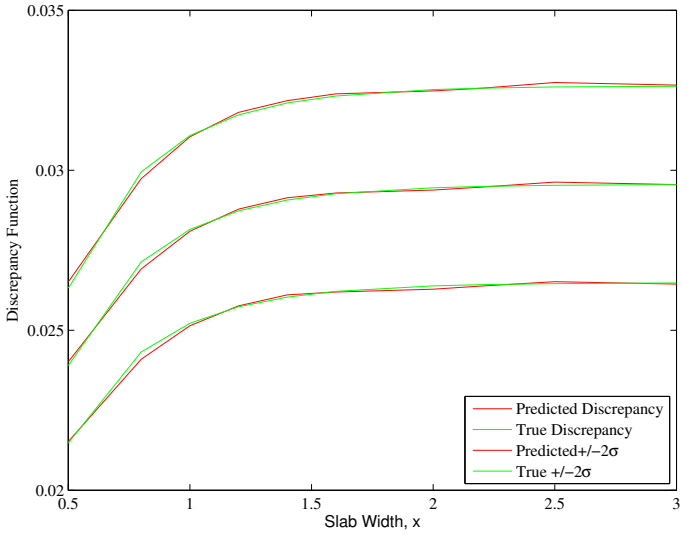


Fig. 2.21. True and predicted model discrepancy term.

The figure includes predictions at testing data of x_{tst}=[1.0,1.4,2.0,2.5]. The true discrepancy is actually a range of the difference between the experimental data and the simulation results at the sampled uncertain inputs; the mean and 2σ values are plotted in the figure. We note that the true and predicted lines are in excellent agreement at these inputs. Figure 2.22 illustrates the full experimental predictions for this test case.

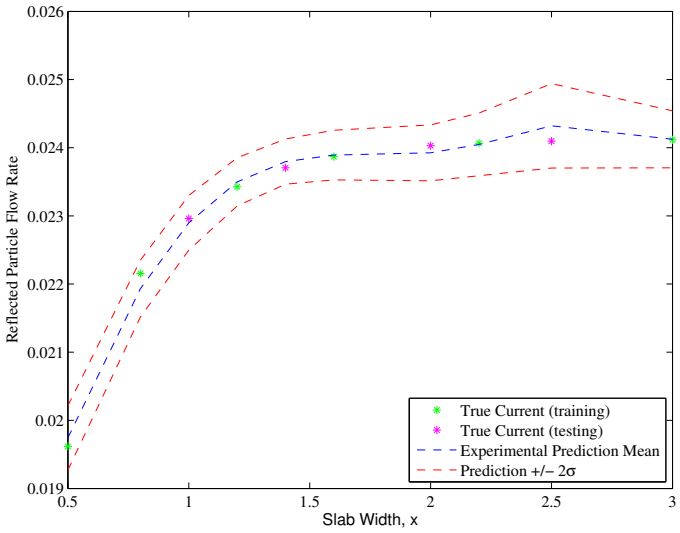


Fig. 2.22. Experimental predictions in the case dominated by the model discrepancy term.

Aside from an overprediction at the slab width of 2.5, we note that the shape of the experimental prediction curves is much more well-behaved than in the case of c=0.99. We note that the rapidly varying behavior at smaller slab-widths is much more dampened, if present at all, in the saturated region, possibly contradicting our notion that the static covariance function is not appropriate for data of this shape.

We believe that this contradiction is explained by the fact that this test case is dominated by the model discrepancy function whereas previous cases were dominated by the emulator. In the GPMSA algorithm, the model discrepancy function is a Gaussian process, but it is not formulated on the field or simulation dataset. Instead it is formulated as the *difference* between the simulation and experimental results. In our case, this difference is better-behaved than the measured/calculated response in previous examples. Also, the simulation response is smoothly varying in the slab-width domain of this example. As a result, each of these functions (the simulation response and model discrepancy) are more appropriately modeled by the

Gaussian-process; therefore, their sum - the experimental response predictions - is more accurate and well-behaved at training points.

2.4 Some Conclusions After Simple Tests of the GPMSA Software

The GPMSA software, derived from the Kennedy and O'Hagan formulation for simulation calibration, is perhaps the most published and common fully-Bayesian implementation of the Gaussian process version of this algorithm. In the previous sections, we set out to test certain portions of its capability in a very simple "universe." In general, we found that the algorithm performs as expected within the capability of the underlying Gaussian processes. For example, a knowledgeable user is allowed to alter priors on the hyperparameters to reflect characteristics of the training data, such as in the case of our introduction of the hidden experimental replicate variable.

We also found, however, that the underlying static covariance function does not respond well to the non-static function being modeled in this "universe". We believe that this example is an exaggerated illustration of the possible benefits to the MMU framework. In a higher-dimensional, more complicated problem, it is not likely that the modeler would be able to recognize as easily the poor predictions being generated by the algorithm. This could lead to inaccurate predictions with possibly dangerous or wasteful consequences. We believe that an exercise of the MMU in a simpler but representative "universe" would help the modeler understand the capabilities, weaknesses, and applicability of a given UQ method in the context of his specific problem.

The final tests further investigated the model discrepancy function. In the first case, there did exist a large discrepancy between the simulation and the experimental results, but the range of the uncertain inputs were large enough that for specific distribution of those inputs (even non-physical values), the simulation replicated the experimental training points. In this case, the algorithm preferred to use the

calibrated uncertain inputs and maintain a near-zero model discrepancy function. In the second test, the range of the uncertain inputs was restricted such that the GPMSA software was forced to include a non-zero model discrepancy function. We found that the software generated a very accurate model discrepancy that resulted in accurate predictions of the experimental response at testing slab widths.

3. ANALYSIS OF THE BMARS ALGORITHM APPLIED TO THE SIMPLE UNIVERSE

As described in Sec. 1.3.2, the Bayesian MARS algorithm uses a specific process to randomly construct an ensemble of regression spline functions that attempt to fit a set of training data. In our implementation, we construct such a function as an emulator for the diffusion simulation. The training inputs are the same as those described in Table 2.1 with the exception that the BMARS emulator does not distinguish between independent and uncertain inputs. Also note that the experimental replicates (varying incident angular flux distributions) have been removed leaving only the "experimental" response to isotropic incident current.

3.1 Choosing the Proper Settings for the BMARS Emulator

The BMARS algorithm allows the user to specify the maximum polynomial order, o, and maximum number of interactions, I, between inputs to the response. In some cases, these choices may be guided by the physical process being modeled. In our case, the analytic solution for the reflected particle flow rate as a function of incident particle flow rate for the diffusion model can be written as:

$$j_{ref} = \left[\frac{\sinh(\frac{x}{L})(1 - 4D\Sigma_a)}{\sinh(\frac{x}{L})(1 + 4D\Sigma_a) + 4\cosh(\frac{x}{L})\sqrt{D\Sigma_a}} \right] j_{inc} \tag{3.1}$$

where x is the slab width and j_{inc} is the incident particle flow rate.

This function does not readily suggest values of o and I, so we can try a range of these parameters to determine the most appropriate BMARS model. Figure 3.1 illustrates the fit of the basis function to the diffusion training data for different values of o and I. A 45° line would be a perfect fit. Because the BMARS algorithm is stochastic, a repetition of each of these constructions would result in a slightly different error.

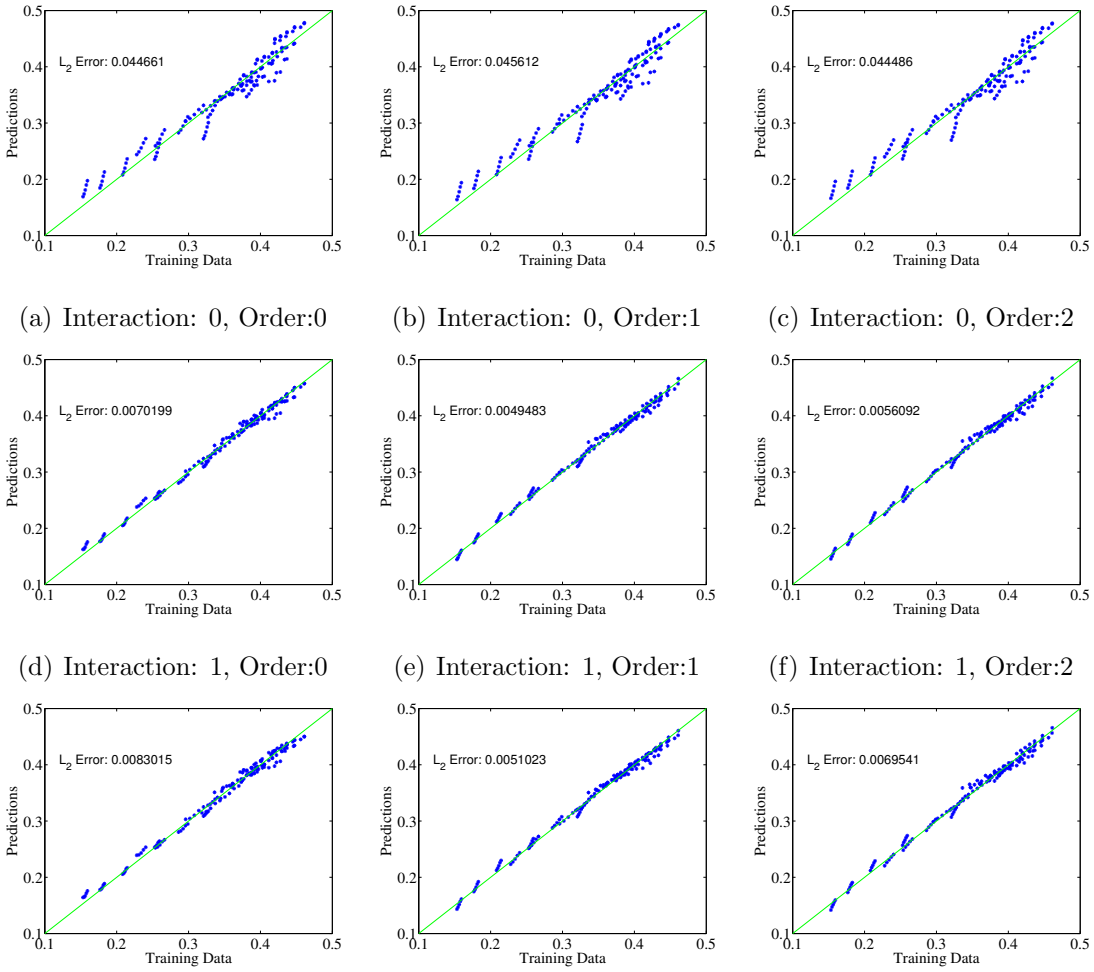
Upon originally proposing the MARS regression idea, Friedman [8] suggested that only linear splines be used in practice, as the effect of higher-order splines can be generated through multiplication of linear splines. Also, in using only linear splines, one avoids the rapid magnification of error in the case of predicting at extrapolated inputs or interpolated inputs that are far from other training data or knot points. Thus, we will allow only linear and constant splines in our BMARS construction. Further, we find that allowing up to 2 interactions (that is, splines that have a component in all three input dimensions) results in very few 3D splines being constructed and accepted. Thus, we will proceed with the following analysis using a maximum of 2D splines.

3.2 Calibrating the Emulator to the Experimental Data

At this point, we have constructed an emulator but have not connected the simulation results to our manufactured experimental data. To do this we will use a filtering and weighting method [13] to calibrate the uncertain inputs to the "experimental measurements".

Here we take advantage of the converged Markov chain process which constructed the BMARS emulator. The realizations of this process can be interpreted as a series of evolving emulators for the simulation data. In the data that follows, the BMARS algorithm was run for 20,000 iterations, and 500 samples were taken uniformly from the last 10,000 of those iterations.

The filtering method allows us to distinguish between the independent and uncertain inputs. If we choose to filter on the uncertain inputs (D and Σ_a), then we randomly sample $n = 1 \ldots N$ points within the convex hull of the training data (in D,Σ_a space). The filter defines a weight for each of the N samples. If the weights are normalized to sum to unity, this can be viewed as a posterior distribution of the input parameters. The filtering method is described as follows:



(a) Interaction: 0, Order:0 (b) Interaction: 0, Order:1 (c) Interaction: 0, Order:2

(d) Interaction: 1, Order:0 (e) Interaction: 1, Order:1 (f) Interaction: 1, Order:2

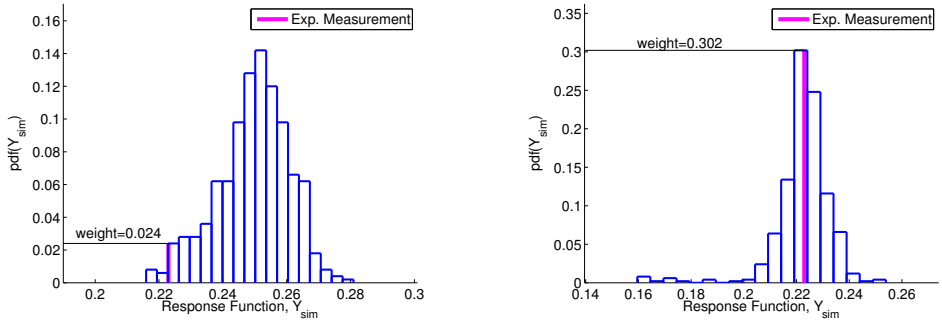
(g) Interaction: 2, Order:0 (h) Interaction: 2, Order:1 (i) Interaction: 2, Order:2

Fig. 3.1. The BMARS fit to the simulation data is most accurate at **I**=1, o=1.

Consider a particular slab width and the n^{th} input point. The 500 randomly chosen emulator functions map this input point to 500 different output values (predictions of the response at this slab width). With each emulator considered equally likely, we construct a discrete probability distribution function for the output value that corresponds to the n^{th} input point. The given input point should receive a

high weight if the discrete pdf is high at the experimentally measured output value and a low weight if the discrtete pdf is low at that value. We define the weight for each point to be the value of the discrete output pdf at the experimentally measured output value.

Figure 3.2 illustrates the assignment of a weight for two different input points, one that receives a low weight and one that receives a high weight. The distribution is simply a normalized histogram of the 500 emulator predictions of the simulation response at a given independent input. The weight of the uncertain input, $\vec{\theta}$ from which this distribution was constructed is simply the value of the pdf evaluated at the experimental measurement for the independent input. In this way, uncertain inputs that result in simulation responses 'close' to experimental responses are given higher weights.



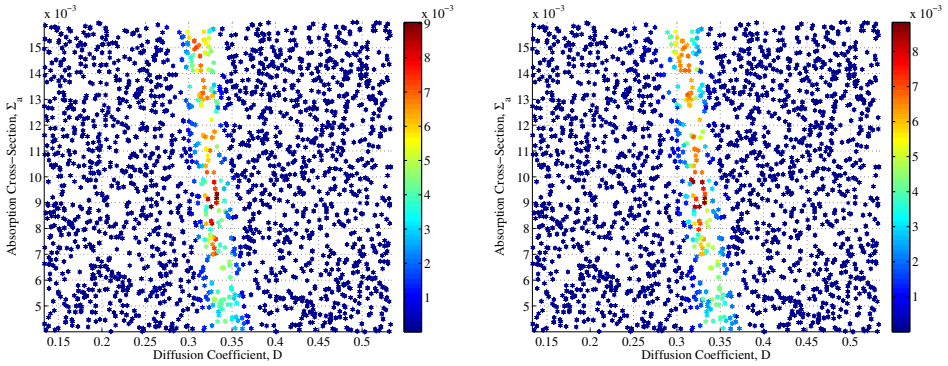
(a) Relatively Low Weight

(b) Relatively High Weight

Fig. 3.2. Samples of the uncertain inputs are calibrated to the experimental measurements.

Note that our scheme of filtering and assigning weights to the uncertain inputs does not assume any functional form for any distribution. For example, we could have assumed that the distribution of emulator predictions was Gaussian and simply assigned a Gaussian posterior distribution function to each component of $\vec{\theta}$. We find no mathematical, statistical, or physical basis for such an assumption and instead choose to assume nothing about the distribution. This strikes us as the best course of action when one can figure out how to achieve it, for it avoids assumptions of symmetry, finite support, etc.

Figure 3.3 plots the weights of the sampled uncertain inputs per slab width. We note a few interesting observations regarding the distribution of the weights. First, the general distribution is somewhat similar for each slab width, and there is a trend of increasing dependence on the absorption cross-section with increasing slab width. For example, for thin slabs, the most likely combinations of the uncertain input mostly depends on the value of the diffusion coefficient, whereas at thicker slabs, the more likely combinations depend on the specific combination of both inputs.



(a) Likelihood Weights, L_s=1cm

(b) Likelihood Weights, L_s=2cm

(c) Likelihood Weights, L_s=4cm

(d) Likelihood Weights, L_s=8cm

(e) Likelihood Weights, L_s=16cm

(f) Likelihood Weights, L_s=32cm

Fig. 3.3. Likelihood distribution of the uncertain inputs at each experimental slab width (red/blue colors correspond to larger/smaller likelihoods).

Another observation is that the distribution of the most likely combinations tends to run off the edge of the domain. This behavior is a result of the continuous dependence of the reflected particle flow rate on the inputs (see Eq. (3.1)). The only exception to this behavior would occur upon filtering values that lead to a local extrema of the function. In this case, we might expect an enclosed distribution of the most likely uncertain inputs.

One technique suggested by these plots is to characterize the movement of the likelihood distribution as the slab width is changed. If we were interested in a prediction at a new slab width, say 12cm, and could estimate the likelihood distribution of the uncertain inputs at 12cm given the likelihood distributions at slab widths surrounding 12cm, we could use the estimated likelihood distribution to generate our prediction. This idea is facilitated by the low dimensionality of this problem; such behavior would be much more difficult to characterize in a more complex "universe". Thus in this thesis, we simply mention this idea but do not attempt an analysis.

We can generate a global probability density function by summing the weights of each uncertain input sample at each experimental slab width and then normalizing to unity. Figure 3.4 illustrates this combined probability density function for the uncertain inputs sampled in Fig. 3.3.

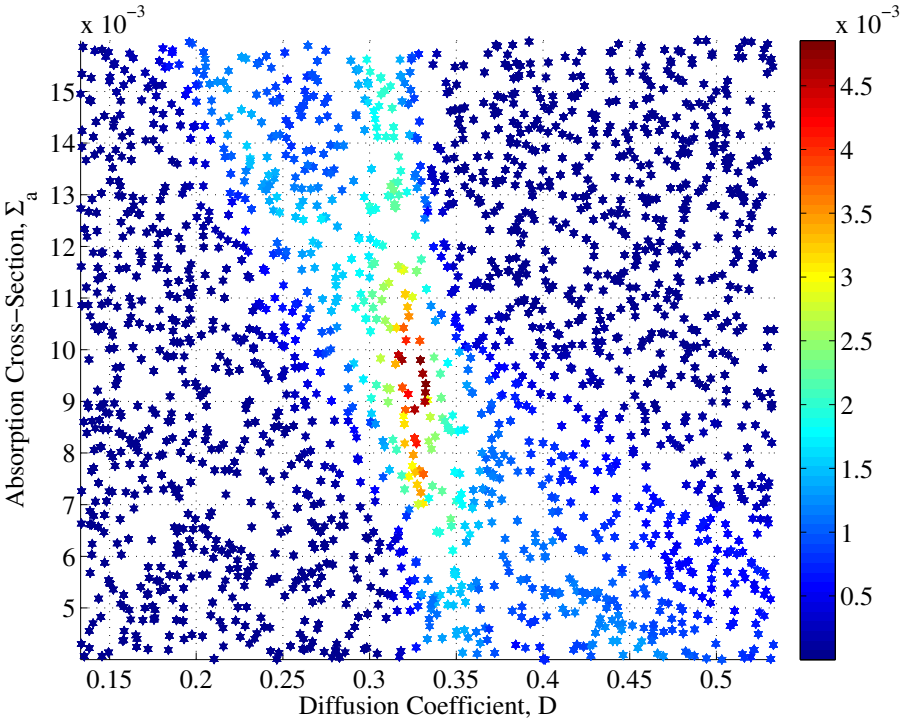


Fig. 3.4. The bivariate combined probability density function for the sampled $\vec{\theta}$ coordinates.

The analysis at this point moves to generating predictions of new experiments using the globally weighted samples of the uncertain inputs.

3.3 Generating Predictions of New Experiments Using the BMARS Emulator

Taking advantage of the relatively low computational costs of the emulator and knowledge gained from the filtering/weighting method, we can attempt to make informed predictions of the reflected particle flow rate at new slab widths. Inherent in this method is the assumption that the emulator is accurately approximating the functional form of the simulator within the convex hull of the training data. To make

an experimental prediction at a new slab width x_i, we simply compute a weighted average of the emulator results:

$$\bar{Y}_{1,predicted}(x_i) = \sum_{n=1}^{N} \eta(x_i, \vec{\theta}_n) w_n$$

where $\eta(x_i, \vec{\theta}_n)$ is the average of the 500 emulators and w_n is the normalized weight of the n$^{\text{th}}$ set of uncertain inputs. If we wished to include a 90% confidence interval on the prediction, then we could use the global likelihood weights to determine the prediction values corresponding to the 5$^{\text{th}}$ and 95$^{\text{th}}$ percentiles. Given a length-N vector $\vec{z}$ and a vector of weights $\vec{w}$ associated with $\vec{z}$, the percentile associated with element n in $\vec{z}$ is simply

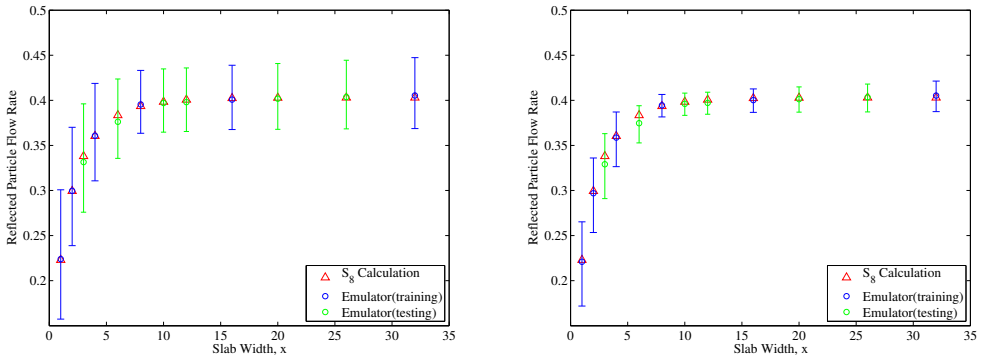
$$p_n = \frac{100}{S_N}\left(S_n - \frac{w_n}{2}\right)$$

where S_n is the partial sum $\sum_{i=1}^{n} w_i$. Then, to find the value associated with the p^{th} percentile, we simply use linear interpolation:

$$z(p) = z_k + \frac{p - p_k}{p_{k+1} - p_k}\left(z_{k+1} - z_k\right)$$

where k is an integer such that $p_k \leq p \leq p_{k+1}$.

Figure 3.5 compares the predictions of this method to a non-weighted prediction (in which w_n=$1/N$). The plot includes testing slab widths of [3,6,12,20,24,28]cm.



(a) Non-Weighted Emulator (b) Weighted Emulator

Fig. 3.5. Assigning weights to the input space improves confidence intervals and slightly increases predictive accuracy.

The immediate observation is that the weighting scheme has the effect of tightening the 90% confidence margins. This behavior can be interpreted as the weighing scheme giving more weight to the combinations of uncertain parameters that result in tight predictions centered about the "true" experimental result. The accuracy of the mean of the emulator predictions is not greatly improved by the weighing scheme. This is likely because our input intervals on the uncertain inputs were centered around the values that produced the best simulation results. It is likely that we would see improvement in the mean emulator predictions if we introduced model error or poor "guesses" at the prior distributions of the uncertain inputs.

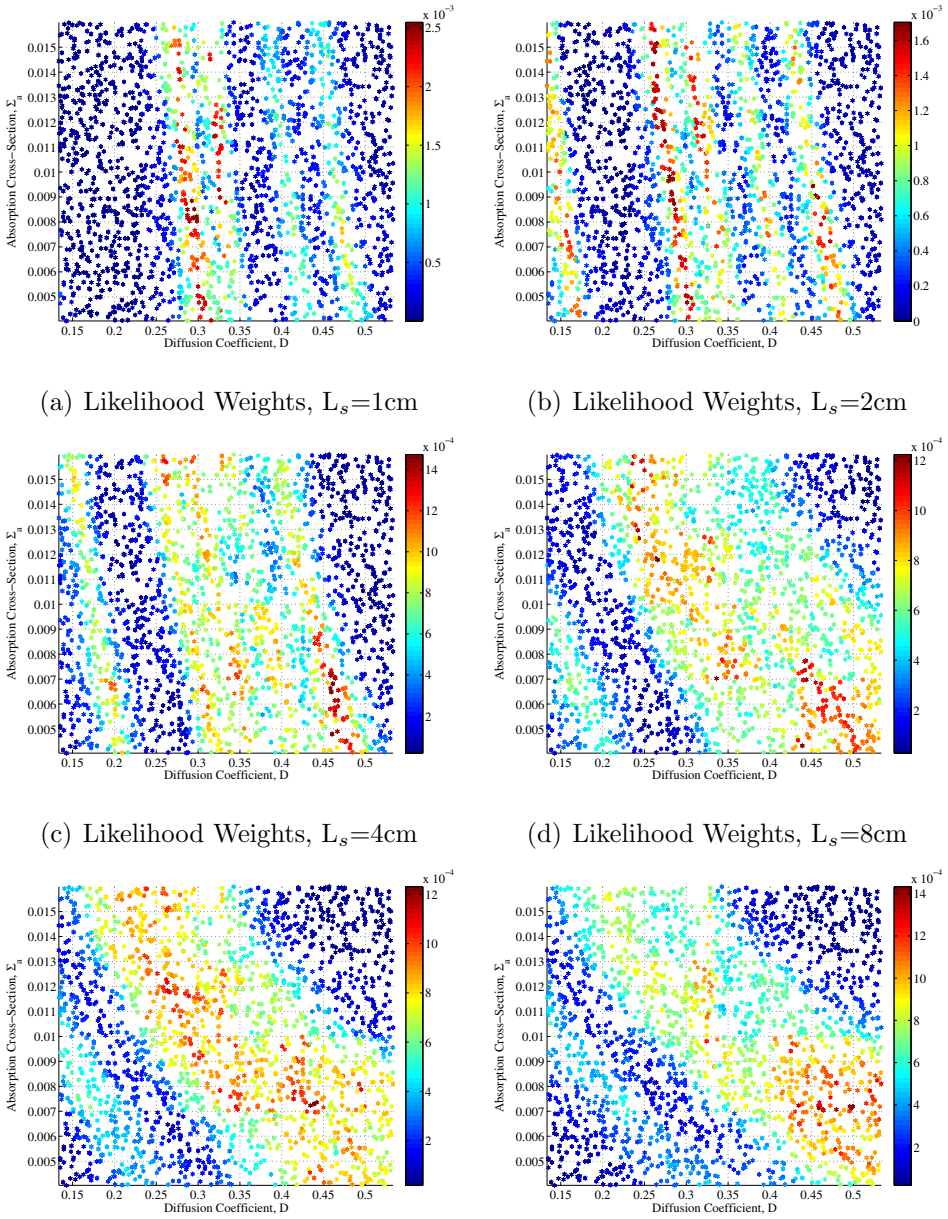
A characteristic of this method that differs from the GPMSA software is that the standard deviation of the predictions at testing points is not larger in magnitude than that at the training data. In general, we expect a higher degree of uncertainty in interpolated or (especially) extrapolated input space as a result of the lack of existing data at these inputs. We find this to be a limitation of this method and plan to study this in future efforts using the MMU framework.

3.4 Demonstration of the Filtering Method in the Presence of Experimental Replicates

We will now show how one might adapt the weighing/filtering method to the case (described in the GPMSA sections) involving multiple experimental measurements at the same input conditions. In the case of this "universe" these experimental replicates are generated by changing the angular distribution of the incidence flux on the slab boundary. The "true physics," S_8 transport method, detects this difference and computes a different response for each replicate. The diffusion approximation, however, does not account for the angular distribution of the incident current and therefore cannot distinguish between the experimental replicates. The input data for this case is illustrated in Fig. 2.12.

The process of constructing a BMARS emulator around the diffusion simulation is the same as described in the previous section. The filtering method also begins as described above, where the uncertain input space is sampled (thoroughly). The assignment of weights, however, must take into consideration the experimental replicates. If we have no reason to consider one set of replicates more "likely" or measured more "accurately" then, as we loop through and construct the discrete pdf of emulator responses at each of the sampled uncertain input coordinates, the weight is computed as the sum of the pdf value at each of experimental response values. These weights are then normalized to unity.

Figure 3.6 illustrates the weight distribution in uncertain input space in this case of multiple experimental replicates.



(a) Likelihood Weights, L_s=1cm

(b) Likelihood Weights, L_s=2cm

(c) Likelihood Weights, L_s=4cm

(d) Likelihood Weights, L_s=8cm

(e) Likelihood Weights, L_s=16cm

(f) Likelihood Weights, L_s=32cm

Fig. 3.6. Likelihood distribution of the uncertain inputs at each experimental slab width with multiple experimental replicates (red/blue colors correspond to larger/smaller likelihoods).

The obvious feature of these scatter plots that is not present in Fig. 3.3 is the multi-modal or streaky nature of the more likely inputs. At thinner slab widths, where the experimental replicates have larger variance, we see more defined streaks across the uncertain input space. On the other hand, at the thicker slab widths, the distribution of the more likely uncertain inputs is more concentrated, but not as concentrated as in the case of a single experimental replicate. Figure 3.7 illustrates the global weights in the case of 5 experimental replicates. Note that the distribution of the weights is much less concentrated than in the case of a single experimental replicate (compare to Fig. 3.4).

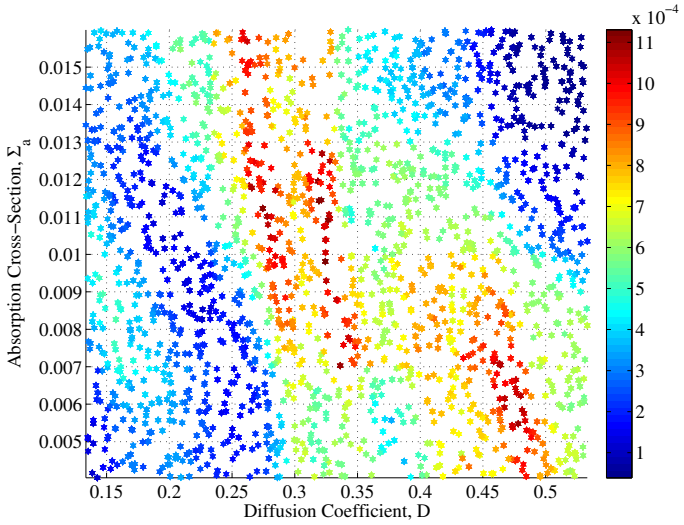


Fig. 3.7. The global bivariate weight distribution in the case of 5 experimental replicates.

We now move ahead with experimental predictions in the same manner as previously described. Because in this case the scattering ratio is very close to 1, we do not include a model discrepancy formulation and generate experimental predictions with a simple weighted average of the BMARS realizations at the uncertain inputs.

Figures 3.8 and 3.9 show experimental predictions from the calibrated simulator and emulator, respectively. Note that we again employ weighted means and percentiles as the illustrative statistics.

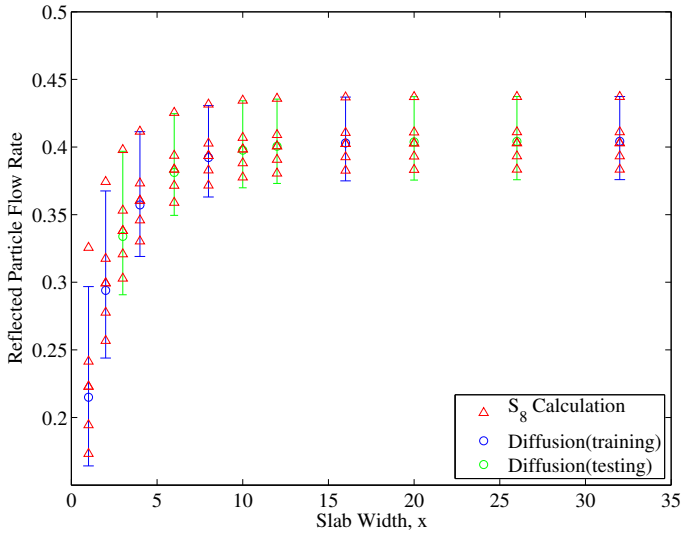


Fig. 3.8. Calibrated diffusion simulator predictions in the case of multiple replicates.

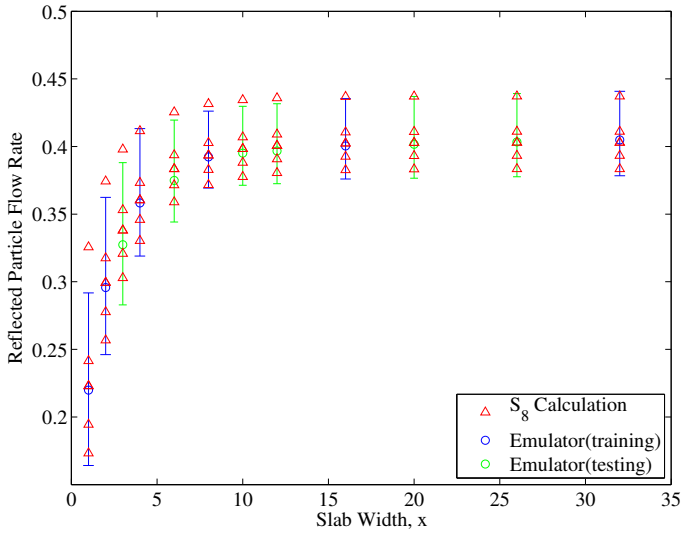


Fig. 3.9. Calibrated diffusion emulator predictions in the case of multiple replicates.

The predictive distributions shown in these figures are encouraging. First, we note very little difference between the calibrated simulator and emulator, indicating that the emulator is in fact closely approximating the simulator, even at testing slab widths. More importantly, however, we note that the 90% confidence intervals nearly enclose all of the experimental observations. An interpretation of the 90% confidence interval is that we would expect one out of every ten true responses to be outside the bounds of the prediction interval. We enclose much more than 90% in each case.

We also note that the results from the BMARS/filtering method in the case of experimental replicates are different and, in some sense, more successful than those from the GPMSA model. Here, the 90% confidence interval seemed to respond to the additional uncertainty imposed by the experimental replicates; that is, the confidence in the experimental predictions decreased. The GPMSA algorithm, however, did not respond to the added uncertainty

3.5 Analysis of the Discrepancy Function Using Ensembles of BMARS Models

We now present an analysis of the discrepancy function similar to that in the GPMSA sections. We return to the Kennedy O'Hagan formulation (see Eq. 1.1) and propose a two step process:

1. Formulate a BMARS emulator of the simulation using available data and calibrate the simulator using the methods described above.

2. Using a certain percentage of the calibrated inputs, formulate a BMARS representation of the model discrepancy function using available experimental data and generate experimental predictions from a combination of the emulator and BMARS model of the discrepancy function.

We will again generate a "universe" with a low scattering ratio such that there exists a large discrepancy between the simulator and experimental results. Figure 3.10 shows the input data for this test (scattering ratio of 0.20).

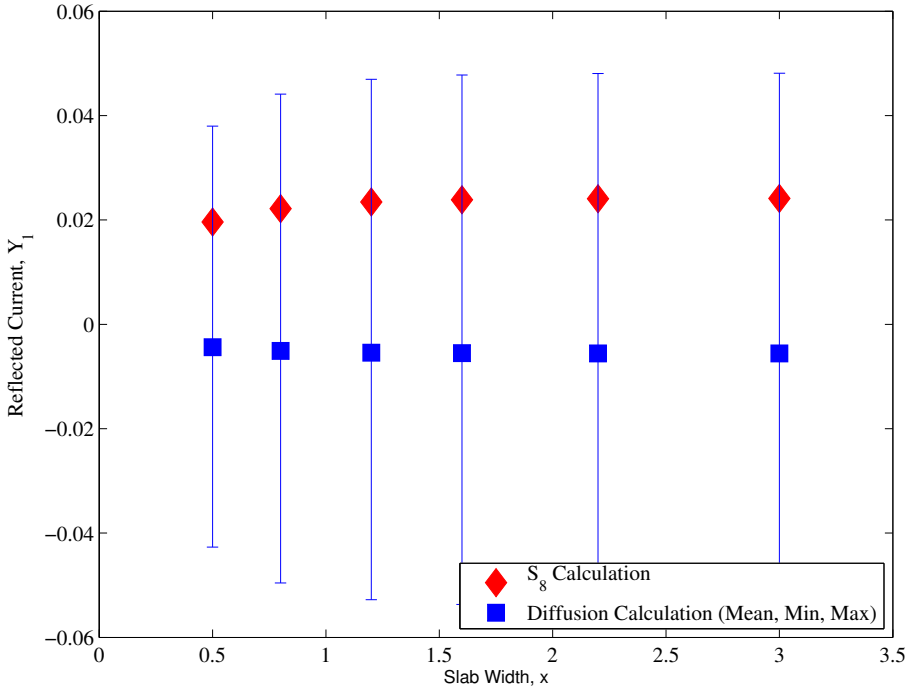


Fig. 3.10. Input data for a scattering ratio of c=0.20.

Following the two-step process, we first construct a BMARS emulator for the diffusion simulation using slab width, diffusion coefficient, and the absorption cross-section as inputs to the model. We then execute the same process as described above for assigning weights to the uncertain inputs, D and Σ_a. At this point, we take advantage of a knowledge of the physics and add a filter to the problem that, for any combination of the uncertain inputs that results in a negative (and therefore non-physical) result, overwrites the existing weight with a weight of zero. This additional filter is not required for the case of c=0.99 because none of the simulation results resulted in negative response functions.

The BMARS emulator was constructed with 10,000 burn-in iterations and 10,000 "converged" iterations using constant and linear splines. Figure 3.11 shows the fit

of the emulator to the simulation results and its relation to the "experimental" measurements:

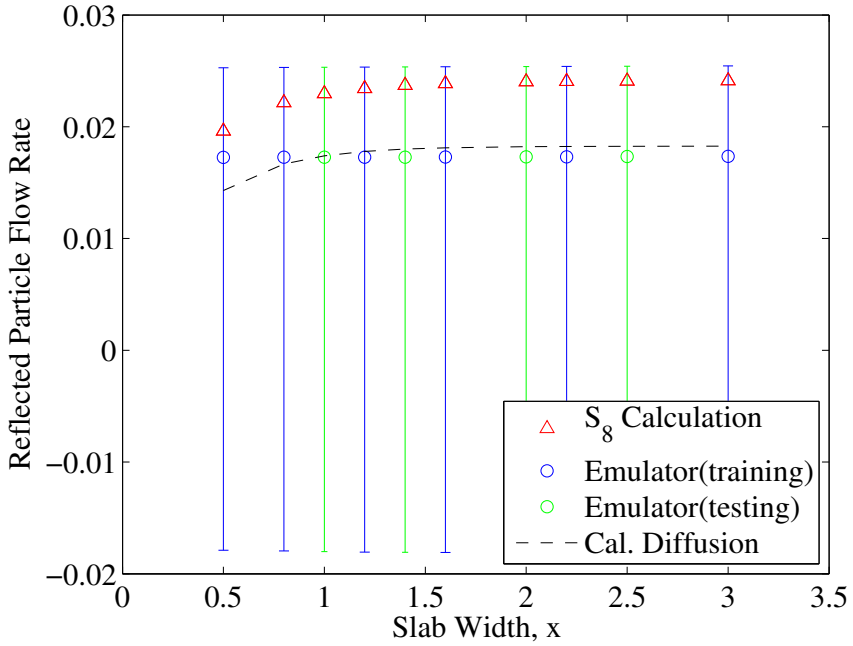


Fig. 3.11. Predictions made from the calibrated emulator and simulator will require a model discrepancy function to replicate the experimental results.

Next we filter and weigh the uncertain inputs based on the experimental predictions. The global results (that is, the normalized sum of the weights assigned at each slab width) is given in Fig. 3.12. It is interesting to compare this set of calibrated uncertain inputs to the range of the same uncertain inputs deemed 'most likely' by the GPMSA software in Fig. 2.17(b).

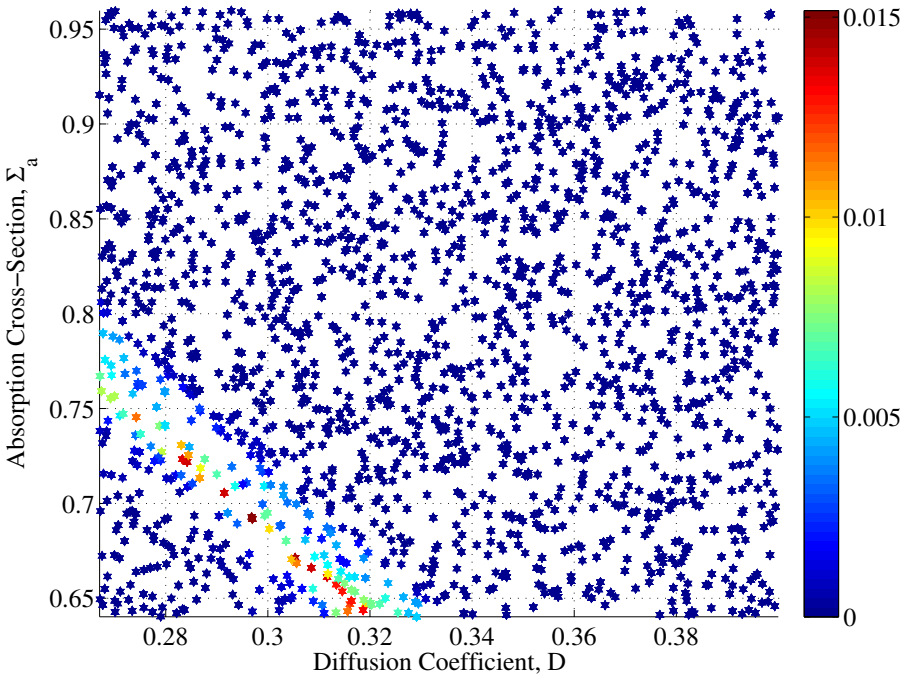
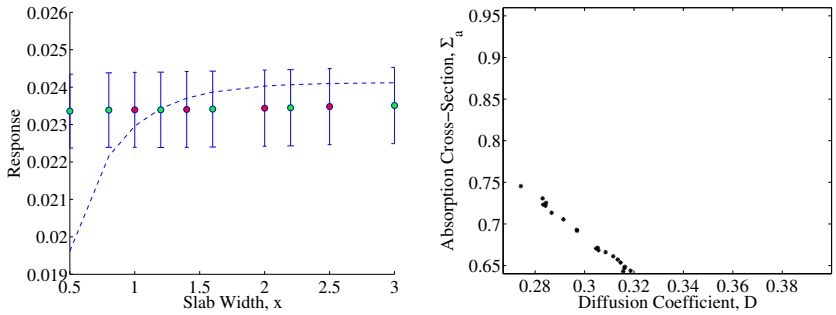


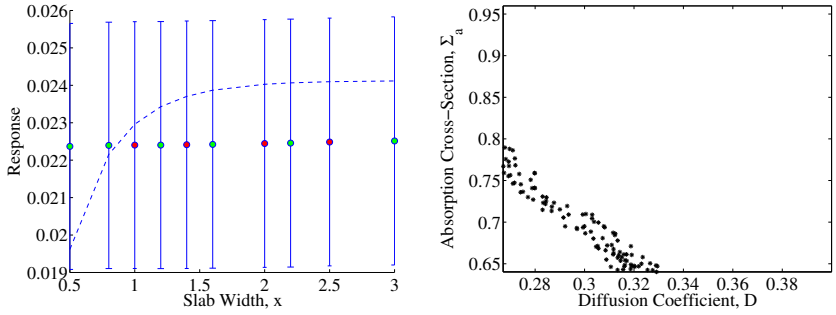
Fig. 3.12. The calibrated uncertain inputs in the case of c=0.20 over all slab widths.

We see that only a small band of uncertain inputs received a weight of appreciable value and that this subset represents a very small percentage of the total input space. To move forward in generating experimental predictions, we must choose an appropriate subset of these uncertain inputs to use in the diffusion calculations. Following the analysis of the previous section, we could use all of the samples in the uncertain input space and compute a weighted average of the emulator response. Alternatively, we could attempt to choose a subset consisting of those uncertain inputs with weights above some threshold or some percentile of those uncertain inputs ranked by weight.

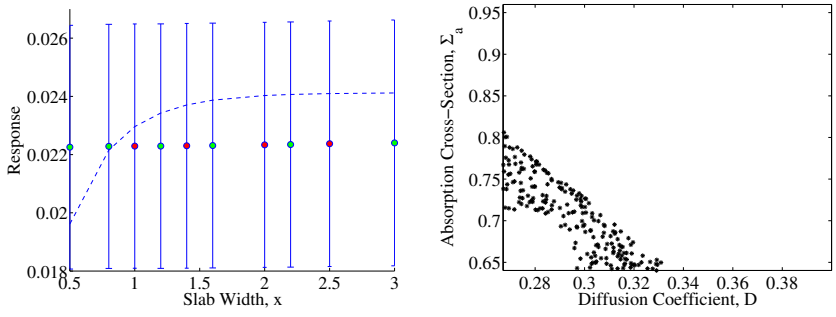
Figure 3.13 illustrates the change in the calibrated simulator and emulator as the number of uncertain inputs used in the calculation is increased. The plots in the left-hand column compare the experimental response to the calibrated (weighted) emulator responses. Following the convention used previously, the green makers indicate training data and the red markers indicate testing data. The emulator predictions are shown as error bars with 90% confidence intervals. The plots in the right-hand column show a scatterplot of those uncertain input coordinates whose global weights were in the 1^{st}, 5^{th}, 10^{th}, and 50^{th} percentiles.



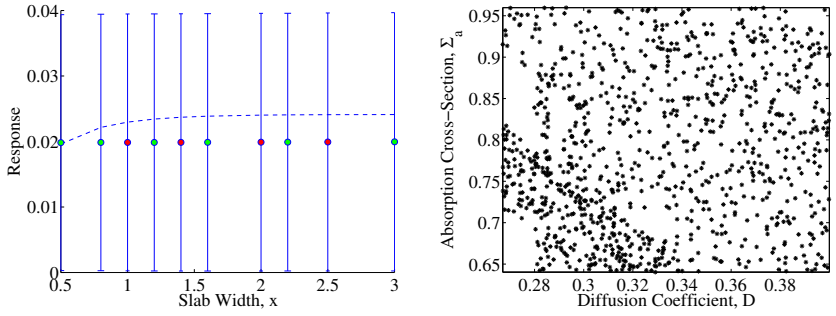
(a) Results with 1^{st} percentile (b) Location of 1^{st} percentile



(c) Results with 5^{th} percentile (d) Location of 5^{th} percentile



(e) Results with 10^{th} percentile (f) Location of 10^{th} percentile



(g) Results with 50^{th} percentile (h) Location of 50^{th} percentile

Fig. 3.13. Location and results from using different percentiles of the weighted inputs.

We see from the plots that the calibrated emulator never seems to take on the shape of the experimental response as a function of slab width. This could be a result of an inaccurate model (diffusion is poorly estimating the response) or an inaccurate regression (BMARS is not properly modeling the response). In either case, if the emulator is to be used in the generation of experimental predictions, there will certainly need to be a model discrepancy function. For the purpose of further illustrating this example, we choose the set of uncertain inputs whose global weights are in the top 10 percentile of the samples.

Following the two-step process previously outlined, we now define a dataset describing the discrepancy function for the generation of a new BMARS emulator. The discrepancy function depends only on the independent input, $\vec{x}$, and is only explicitly calculable at existing experimental data points. We generate the discrepancy model training data by subtracting the weighted mean emulator prediction from the existing experimental measurements (note that we can do this for training and testing data). Figure 3.14 illustrates the relationship between the mean emulator response, experimental measurements, and calculated discrepancy function at the experimental training data.

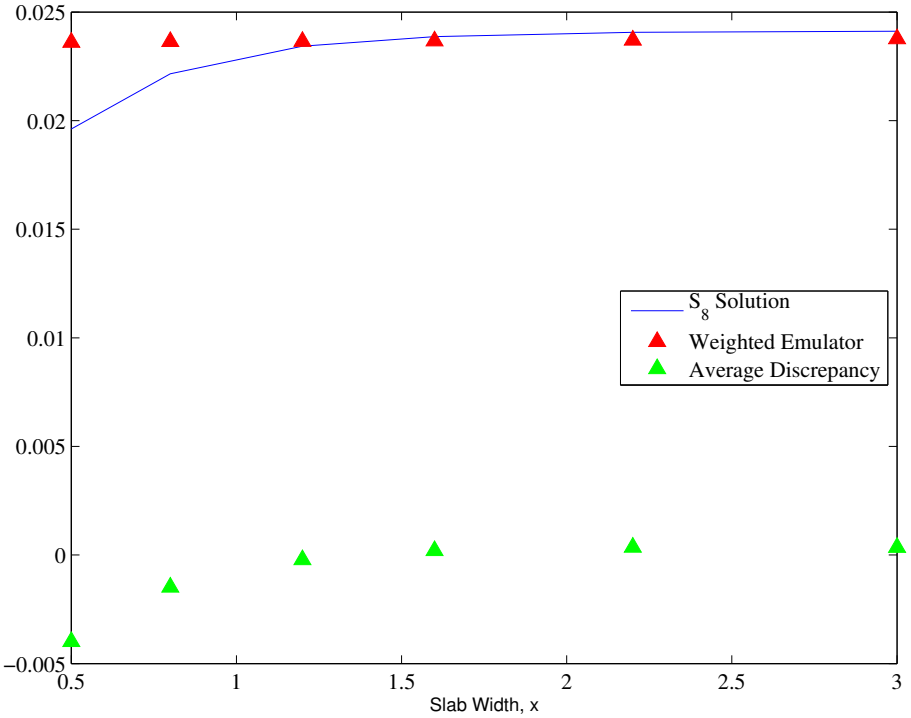


Fig. 3.14. Training data for the BMARS discrepancy model.

Figure 3.15 shows a BMARS fit to the discrepancy function for both training and a set of testing data. The predictions show a 90% confidence interval generated from 500 posterior samples from the MCMC chain used to generate the BMARS basis function.

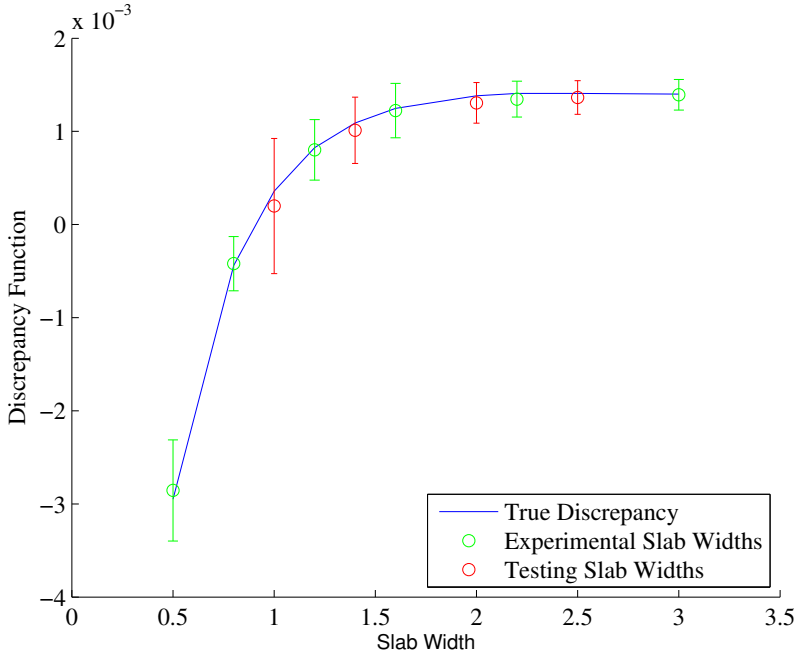


Fig. 3.15. Predictions of the discrepancy function for a range of BMARS sample sizes.

After generating the surrogate BMARS model for the discrepancy function, we simply add results from the two BMARS models to generate predictions. The mean prediction at a given slab width is simply the sum of the means from the two emulators. The reported error is the quadrature sum of the standard deviation of the emulator and discrepancy BMARS models. Figure 3.16 illustrates the experimental predictions.

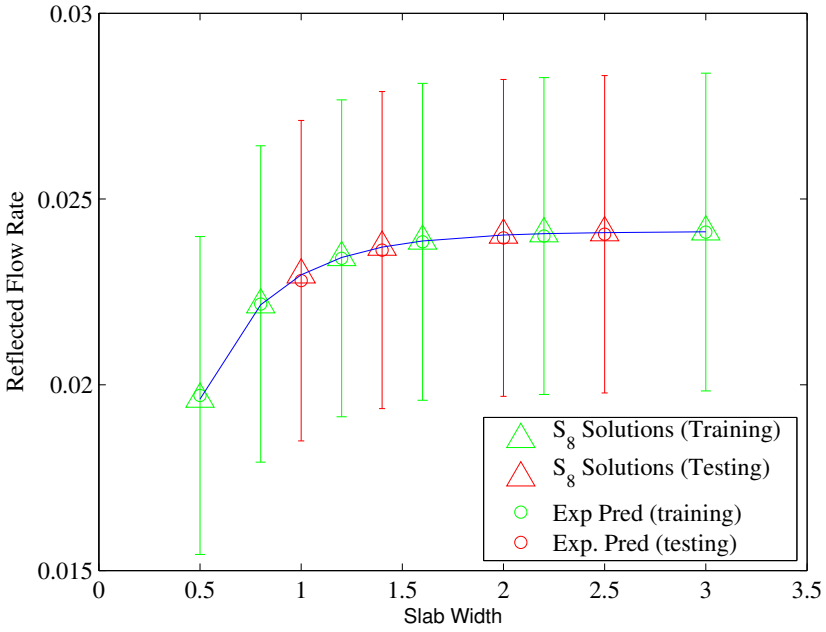


Fig. 3.16. Experimental predictions in the case of a model discrepancy.

The expected value of the experimental predictions for both training and testing slab widths are very accurate. The 90% confidence interval is dominated by uncertainty in the emulator as opposed to uncertainty in the model for the discrepancy function. Recall that the top 10% weighted uncertain inputs were used to generate the weighted emulator predictions and therefore the discrepancy function model. In the specific case of this problem, the use of these 10% includes a number of $\vec{\theta}$ coordinates with very low weights (compare Fig. 3.12 with Fig. 3.13(f)). The result of doing so is a large increase in the predictive variance of the BMARS model (compare Fig. 3.13(a) to Fig. 3.13(e)).

To reduce the predictive variance, we could use only the top 1% weighted uncertain input coordinates. The resulting experimental predictions are given in Fig. 3.17:

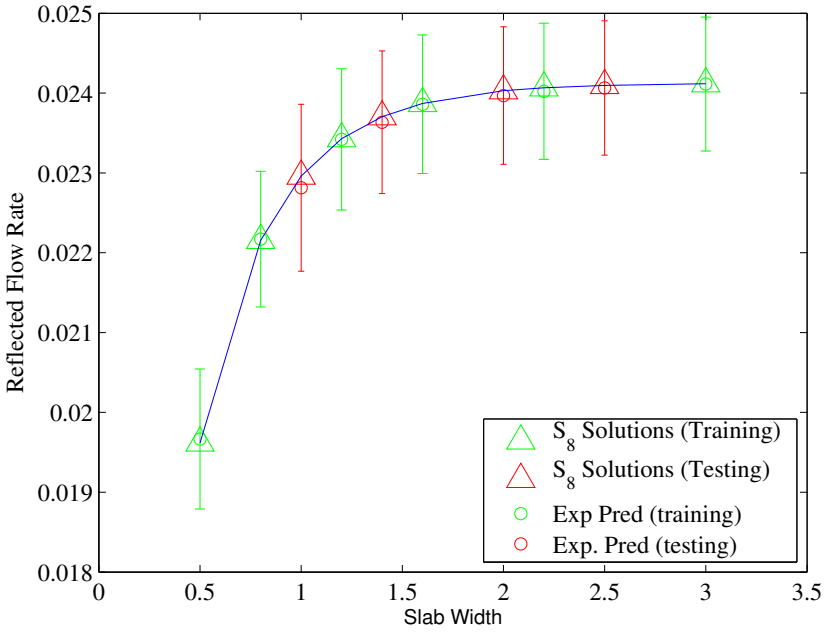


Fig. 3.17. Decreased predictive variance resulting from a more refined sample of uncertain inputs.

We see a decrease in the predictive variance because the variance in the emulator is reduced. The appropriateness of the reduction to just 1% of the uncertain inputs will depend on the problem, the ability of the modeler to interpret the physical implications of the tight restriction in $\vec{\theta}$ space, and the total number of uncertain input samples taken to begin with. In our case, the low dimensionality of $\vec{\theta}$ space allowed for fairly dense sampling. Further, the results of the calibration were validated in the sense that the GPMSA software returned similar posterior distributions. However, in a real-world problem of interest, this kind of reduction may be unrealistic or against physical intuition.

3.6 Some Conclusions After Development of the BMARS/Weighing/Filtering Method

The previous sections outlined an extremely simple implementation of model calibration using a weighing and filtering method. We first showed that, given a sufficient number of simulator runs and a relatively accurate computational model (*i.e.*, a discrepancy function is not required), a modeler can use this filtering method to identify a region in the uncertain input space that is more likely to lead to accurate experimental predictions using the simulator alone. In this case, the BMARS algorithm was used to fit the simulation data, and the distribution of the predictions from the basis function was used to assign a weight to each sampled uncertain input coordinate. These weights are then used to generate experimental predictions using a simple weighted average of the response functions.

In our highly-scattering particle transport "universe," the construction of the emulator allowed for an accurate fit to (and thus accurate predictions of) the simulator. Our filtering/weighting scheme identified a volume of uncertain input coordinates near the "true" value of these inputs, and this volume was very similar to the calibrated distributions produced by the GPMSA software. Finally, we showed that the confidence interval of experimental predictions via the calibrated simulator are smaller than the uncalibrated simulator. In other words, by calibrating the uncertain inputs, we are able to increase the confidence in our experimental predictions.

When we extended the highly-scattering "universe" to include 5 experimental replicates per slab width, our BMARS emulator and filtering method responded by identifying multiple volumes of calibrated inputs and thereby increased the 90% predictive confidence intervals for the experimental response. This response was in contrast to that of the GPMSA software, which did not seem to respond to the experimental replicates without modeler action.

In the case of a less accurate model requiring a discrepancy function, we proposed and implemented a two-step process whereby a BMARS model is first built to emulate

the simulator. If possible, a likely volume of the uncertain inputs is identified via the same filtering/weighting method described above. We then estimate the discrepancy function by simply subtracting the mean emulator response at experimental slab widths from the experimental measurement at those slab widths. This discrepancy is then fit with another BMARS model, which can then be combined in the Kennedy O'Hagan fashion to generate experimental predictions.

Although this problem contained only two uncertain inputs, we foresee a simple extension to higher-dimensional problems. The low computational costs of BMARS and most other response models does not prohibit dense sampling of the uncertain input space. The modeler should keep in mind that, even if the costs of sampling the response surface becomes non-negligible, it is still considerably less than the cost of running the simulation.

4. CONCLUSIONS AND DISCUSSION

We can make a number of conclusions regarding the application of these UQ methodologies to our simple universe. Our experience with the GPMSA software was a showing example of the possible consequences associated with "black-box" usage of statistical models. Our initial unfamiliarity with the Bayesian formulation and the Gaussian process model led to our use of default parameter values and prior distributions, which we found were not appropriate for some perturbations of our universe. In the end, the MMU framework facilitated added understanding of the GPMSA methods, but we still found that the stationary GP model was not appropriate for our functional output. The BMARS emulator, on the other hand, tended to interpolate our data more accurately, and we showed through a filtering method that we could improve confidence in our experimental predictions by calibrating the uncertain inputs.

We further tested the flexibility of the UQ methods by imposing sources of uncertainties into our simple universe. In the case of the GMPSA algorithm, we found that the software did not account for the addition of "experimental noise" (resulting from experimental replicates or an imposed "measurement" error) as we expected. This unexpected behavior facilitated an exercise of the MMU framework: we learned how to tune the priors of the GP hyperparameters for our problem of interest and can apply this added knowledge to our more complex problem.

In the case of BMARS, uncertainty resulting from experimental replicates was handled more naturally by the filtering/weighting method described in this thesis. The additional "experimental measurements" resulted in a more uniform distribution of the global weights, which translated into larger confidence intervals in the experimental predictions. In other words, we are less confident in the most likely distribution of the uncertain parameters; therefore, we have larger uncertainty in the predictions made by our simulator.

Finally, our development of a Kennedy/O'Hagan-like implementation of an emulator/discrepancy model using BMARS resulted in accurate experimental predictions in the case of an inaccurate simulator (that is, the diffusion approximation to a highly-absorbing medium). We found that calibration of the uncertain inputs in the case of an inaccurate simulator results in only a small volume of uncertain inputs with appreciable weights. Further, the choice of uncertain inputs to use for approximating the simulator response and generating a discrepancy function affects the predictive variance in experimental response at new slab widths. The MMU framework allowed us to explore these ideas and further understand how BMARS may be applied in a larger problem.

We can also make a number of more general conclusions resulting from our first application of MMU. First, especially in the case of GPMSA, the simplicity of the "universe" effectively facilitated added understanding of the UQ model (which is exactly what MMU is designed to do). Therefore we can conclude that the modeler must carefully balance a simple, controllable universe with one that properly reflects the complexity of the real world problem. Second, many of the statistical assumptions and formulations within the GPMSA and BMARS algorithms are unfamiliar to us (a group of physicists); in fact, we concede that further optimization of the GPMSA algorithm could certainly improve its predictive capability for our problem. Thus, we make the strong recommendation that any modeler with limited initial understanding of a statistical UQ formulation work *very* closely with an expert (perhaps mathematician or statistician) in the methods used to drive the uncertainty quantification. Correctly understanding, accounting for, and applying the statistical models is a requirement for their accurate treatment of uncertainty in the system.

We can further conclude that the utility of an MMU study will be governed by the modeler's ability to design a universe that properly represents a real-world problem of interest. The coupling between physics and uncertainty quantification is necessarily problem dependent. The proper choice of manufactured reality and approximate

model, inputs to those models, boundary conditions, sampling points, and imposed uncertainties will facilitate the transfer of knowledge gained in an MMU study to the real-world problem. A properly designed universe will also give the modeler a chance to develop improvements for and exploit degrees of freedom within the UQ methodology such that the coupling between that methodology and the physics of interest is further enhanced.

REFERENCES

[1] M.C. Kennedy and A. O'Hagan. Bayesian calibration of computer models. Journal of the Royal Statistical Society (Series B) 63(3) (2001) 425-450.

[2] D. Higdon, J. Gattiker, B. Williams, and M. Rightley. Computer model calibration using high dimensional output. Journal of the American Statistical Association 103 (2008) 570-583.

[3] D. Higdon, M. Kennedy, J.C. Cavendish, J.A. Cafeo, and R.D. Ryne. Combining field data and computer simulations for calibration and prediction. SIAM Journal of Scientific Computing 26 (2004) 448-466.

[4] K.M. Case and P.F. Zweifel. Linear Transport Theory. Addison-Wesley, Reading, MA, 1967.

[5] A.E. Maslowski and M.L. Adams. Behavior of continuous finite element discretizations of the slab-geometry transport equation, in Proceedings of the Conference of Mathematics and Computation, Supercomputing, Reactor Physics and Nuclear and Biological Applications, Avignon, France, September 2005. CD-ROM.

[6] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of state calculations by fast computing machines. The Journal of Chemical Physics 21(6) (1953) 1087-1091.

[7] W.K. Hastings. Monte carlo sampling methods using markov chains and their applications. Biometrika 57(1) (1970) 97-109.

[8] J.H. Friedman. Multivariate adaptive regression splines. The Annals of Statistics 19(1) (1991) 1-67.

[9] D.G.T. Denison, B.K. Mallick, and A.F.M. Smith. Bayesian mars. Statistics and Computing 8(4) (1998) 337-346.

[10] J.R. Gattiker. Using the gaussian process model for simulation analysis (gpm/sa) code. Unpublished Documentation from Los Alamos National Laboratory.

[11] C.L. Paclorek and M.J. Schervish. Spatial modelling using a new class of nonstationary covariance functions. Environmetrics 17(5) (2006) 483-506.

[12] V.R. Joseph, Y. Hung, and A. Sudjianto. Blind kriging: A new method for developing metamodels. Journal of Mechanical Design 130(3) (2008) 31-102.

[13] P. Dykema, S. Brandon, B. Johnson, and G. Johannesson, January 2010. Personal Communication. Lawrence Livermore National Laboratory.

VITA

Hayes Franklin Stripling IV received his Bachelor of Science degree in May of 2009 from the Nuclear Engineering Department at Texas A&M University in College Station, Texas. Thereafter, he entered the graduate program in nuclear engineering at the same university and received a Master of Science degree in December of 2010. His research interests are uncertainty quantification in scientific computing, verification, validation, and calibration of computer models, and the design and analysis of massively-parallel multi-physics calculations.

Hayes is a Department of Energy Computational Science Graduate Fellow. As a part of this program and through collaboration with his advisors, Hayes has held summer research positions at Knolls Atomic Power Lab in Niskayuna, NY, Sandia National Laboratories in Albuquerque, NM, and Lawrence Livermore National Laboratory in Livermore, CA. His hobbies include following Texas A&M baseball, traveling, hiking/camping, and spending time with his family.

Hayes may be reached at the following address: Nuclear Engineering Department; Texas A&M University - 3133 TAMU; College Station, TX, 77843-3133. His email address is h.stripling@tamu.edu.