A COMPRESSED SENSING APPROACH TO UNCERTAINTY PROPAGATION FOR APPROXIMATELY ADDITIVE FUNCTIONS

A Thesis

by

KAIYU LI

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

MASTER OF SCIENCE

Chair of Committee,	Douglas Allaire
Committee Members,	Raktim Bhattacharya
	Richard Malak
Head of Department,	Andreas A. Polycarpou

December 2016

Major Subject: Mechanical Engineering

ABSTRACT

Computational models for numerically simulating physical systems are increasingly being used to support decision-making processes in engineering. Processes such as design decisions, policy level analyses, and experimental design settings are often guided by information gained from computational modeling capabilities. To ensure effective application of results obtained through numerical simulation of computational models, uncertainty in model inputs must be propagated to uncertainty in model outputs. For expensive computational models, the many thousands of model evaluations required for traditional Monte Carlo based techniques for uncertainty propagation can be prohibitive. This paper presents a novel methodology for constructing surrogate representations of computational models via compressed sensing. Our approach exploits the approximate additivity inherent in many engineering computational modeling capabilities. We demonstrate our methodology on some analytical functions, with comparison to the Gaussian process regression, and a cooled gas turbine blade application. We also provide some possible methods to build uncertainty information for our approach. The results of these applications reveal substantial computational savings over traditional Monte Carlo simulation with negligible loss of accuracy.

ACKNOWLEDGMENTS

I would first like to thank my advisor Dr. Douglas Allaire at Texas A&M University, who has supported me throughout my thesis with his patience, enthusiasm and vast knowledge. I simply could not imagine having a better advisor for my Masters study.

I would also like to thank my committee members, Dr. Raktim Bhattacharya and Dr. Richard Malak for their encouragement and passionate participation.

I would like to thank all my fellow labmates in Computational Design Laboratory at Texas A&M University for their friendly advice and academic help. And I would like to thank Dr. Kenneth Bryden and Zachary Reinhart at Iowa State University for enlightening my first entry to the field.

Last but not the least, I would like to thank my parents for their love and continuous support throughout my years of study. This accomplishment would not have been made without them.

NOMENCLATURE

ECDF	Empirical cumulative distribution function
GP	Gaussian process
HDMR	High-dimensional model representation

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION AND BACKGROUND *

1.1 Motivation

Increasingly, computational models of physical systems are used to support decision-making processes [1]. These processes include design decisions, policy level analyses, and experimental design decisions among other things. Typically, computational models have uncertainty associated with their inputs and parameters, which subsequently propagates through the model, begetting uncertainty in the outputs, or quantities of interest, the model estimates. Therefore, the application of model output information to support decision-making processes requires the effective propagation of uncertainty. The process of propagating uncertainty from model inputs to model outputs could be conducted, for example, via Monte Carlo simulation, but this approach traditionally requires several thousand model evaluations. However, computational models of physical systems of practical use in engineering decision-making processes are often expensive in terms of the time it takes to compute a set of outputs given a set of inputs. Therefore, conducting several thousand model evaluations for something like Monte Carlo simulation is often computationally intractable.

1.2 Research objective

For the case of computationally expensive models, recourse is often made to surrogate modeling techniques, both through approximations of the underlying physics of the modeling capability, and the underlying uncertainty of the model inputs. We want to present a novel surrogate modeling approach for uncertainty propagation constructed from the powerful concept of compressed sensing. Given a functional basis representation for which

*Reprinted with permission from "A compressed sensing approach to uncertainty propagation for approximately additive functions" by Kaiyu Li, Douglas Allaire, 2016. ASME 2016 International Design Engineering Technical Conferences, IDETC/CIE, Copyright 2016 by ASME.

the output of a computational model is sparse, we would like to demonstrate that we can recover near-exact approximations of the computational model as a function of its inputs with remarkably few function evaluations. The surrogate representation of the model we proposed should be used with any supported probability distribution to provide rapid uncertainty propagation results. We would like to make a comparison between our method and Gaussian Process (GP) regression method to see how well our method could perform. Based on the computational model we recover, we also want to discover some possible methods to build uncertainty information for our method.

However, our approach is specifically catered to the class of what we refer to as approximately additive functions. By this, we mean functions that depend primarily on additive terms of subfunctions of low dimension. For example, a function of many inputs maybe adequately represented by a sum of subfunctions of each individual input and a few subfunctions of two input combinations (i.e., interaction terms). As discussed in the thesis, this approximately additive characteristic is believed to be a feature of a wide-variety of computational models often used for engineering purposes.

1.3 Background

Uncertainty propagation, in the context of computational modeling, involves mapping the uncertainty in the inputs to uncertainty on the outputs of the computational model. Without loss of generality, we will deal with computational models that yield a single output, or single quantity of interest. The process of mapping uncertainty from inputs to model output often involves a sample-based approach to gathering input settings that are then either mapped through the full computational model by evaluating the model output at each input setting, or are mapped through an approximation, or surrogate model, of the computational model.

The most basic and most common approach to propagating uncertainty through com-

putational models is via Monte Carlo simulation. It is a widely used numerical methods in aspects like: finite mixture distribution [2], system reliability and risk analysis [3] and uncertainty analysis [4]. For a general computational model, $f(\boldsymbol{X})$, where $\boldsymbol{X} = (X_1, X_2, ..., X_d)^T$, and the $\boldsymbol{X}$ is a random vector, Monte Carlo simulation works by sampling a point $\boldsymbol{x}$, from the distribution of $\boldsymbol{X}$, and then running the computational model to evaluate $f(\boldsymbol{x})$. If this process is repeated several times (usually thousands), then the strong law of large numbers guarantees that the empirical distribution of the output evaluations converges almost surely to that of $f(\boldsymbol{X})$ [5]. That is

$$\hat{F}_n(\boldsymbol{t}) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_i(\boldsymbol{t}) \xrightarrow{a.s.} F(\boldsymbol{t}) \text{ as } n \to \infty, \tag{1.1}$$

where $\hat{F}_n(t)$ is the empirical distribution of $f(\boldsymbol{X})$ generated by sampling, F is the cumulative distribution function of $f(\boldsymbol{X})$, $\mathbf{1}_i(\boldsymbol{t})$ is the indicator function for the event, and $x_j^i \leq t_j, \forall j \in \{1, 2, ..., d\}$, where x_j^i is the i-th sample of the j-th input. Given this convergence property, Monte Carlo simulation, with enough input samples, is often considered as the golden standard to compare against when developing new algorithms for uncertainty propagation. It has also seen wide use in many engineering and scientific applications [6, 7, 8, 9, 10, 11, 12]. However, the convergence rate, which is governed by the central limit theorem, is $O(1/\sqrt{n})$, thus making the method impractical for most expensive computational models for uncertainty propagation [13]. The limiting factor in aspect of standard Monte Carlo method accuracy is that the clumping could occur in the random sequences or pseudo-random sequences. Clumping means that some points could lie very close together. Clumping could occur because the points are independent in the sequences as the points do not know anything about each other.

To improve upon the convergence performance of traditional Monte Carlo simulation, different sampling strategies have been studied and applied, such as quasi-Monte

Carlo method, importance sampling, Latin hypercube sampling and stratified sampling [14]. Quasi-Monte Carlo method uses quasi-random (also known as low-discrepancy) sequences instead of random or pseudo-random sequences, and it is designed to fill the input space as uniform as possible (with low discrepancy) [15]. The elements of a quasi-random sequence are correlated, which make them more uniform than the random sequences [16]. The convergence rate of Quasi-Monte Carlo is $O((\log n)^d/n)$, where d is the number of inputs, and is thus generally superior compared to the traditional Monte Carlo method when d is not too large. However, quasi-random sequences are less versatile than random or pseudo-random sequences due to correlations between points. They are designed for purpose like integration, rather than simulation or optimization purposes. As the desired result of a simulation can often be written as an integral, quasi-Monte Carlo is then being applied. However, this method is often involved with high dimensionality, which could limit the effectiveness of quasi-Monte Carlo sequences [16].

Importance sampling method is used for understanding distribution properties when only the proposal distribution is known. Importance sampling weight is constructed to pick importance values of random input variables to correctly estimate target distribution properties. It is recently used in feed-forward multi-component systems uncertainty analysis [10] and reliability calculations [17]. However, it could be hard to implement efficiently due to multiple solutions for different problem classes in a general framework.

Latin hypercube sampling, or LHS [18] operates by sampling on what is referred to as a Latin hypercube, which requires each sample to be the only sample in each hyperplane it belongs to in the hypercube. Stratified sampling proceeds by placing samples in certain subsets, or strata, in a sample space, and then weighting these points by the probability of being in that particular stratum. Convergence rates of Latin hypercube and stratified sampling methods tend to be superior to Monte Carlo simulation for low input dimension, but degrade as the number of inputs increase.

For expensive computational models, the uses of sample based approaches that use the full model are often computationally prohibitive. A typical strategy in these circumstances is to create a surrogate or metamodel of the full model using a small set of samples from the full model. Often data-fit techniques, such as response surface modeling or the use of Gaussian processes are employed to construct such models [19, 20, 21]. Gaussian process regression is a fully probabilistic method that is flexible and could be implemented easily. Gaussian process regression is widely used in problems like: nonlinear signal processing [22] and Bayes Filters [23], and localization sensor network uncertainty [24]. Gaussian process regression provides fully probabilistic predictive distributions and knowledge of kernel parameters, but it has a high computational cost, which grows as $O(n^3)$ [25].

Other alternatives are hierarchical surrogate models [26, 27] that consider hierarchies of modeling assumptions or computational grids, and reduced-order models [28] that rely on underlying knowledge of the governing equations of the computational model, such as polynomial chaos expansion and anchored ANOVA (i.e. cut-HDMR (high dimensional model representation)). The result of developing a surrogate model is usually a much cheaper, in a computational sense, version of the computational model that can be used for various purposes, such as uncertainty propagation. The reduction in runtime, however, usually comes at the expense of some loss in accuracy, or fidelity, in the surrogate model.

Polynomial chaos expansion is a useful technique for propagating uncertainty through partial and ordinary differential equations. Random variables are expanded in terms of orthogonal polynomials, and the expansion coefficients could be derived from differential equations [29]. Polynomial chaos expansion methods have found increasingly used in the probabilistic uncertainty quantification field over the past decade [30]. Polynomial chaos expansion method could efficiently represent and propagate complex models with large uncertainty. However, it is strongly dependent on the tails and the shape of the objective distributions. Polynomial chaos expansion for uncertainty qualification is widely used in

areas like: aeroelastic modeling, chemical reactors and switching systems [31].

The anchored ANOVA (Analysis of variance) method has recently been used in many literatures. Specifically, it is proposed in [32, 33] to decompose statistical moments, which is based on the covariance decomposition of output variance. The computational cost of deterministic simulations is considerably reduced to get accurate results, and it is less sensitive to the choice of the anchor point. However, it is hard to build the surrogate model as the decomposition of higher order component functions increases exponentially in aspect of the dimension of deterministic simulation model, and this increases the computational cost.

Another alternative to deal with an expensive computational model in uncertainty propagation is the use of a surrogate representation of the uncertainty in the inputs. Some techniques for this include implicit uncertainty propagation, moment matching, the advanced mean value method, and unscented transforms [34, 35, 36, 37, 38, 39]. Moment based propagation method is simple to implement and it is widely used in complex systems. Moments (i.e. mean, variance, skewness, kurtosis, and so on) are random variables efficient descriptors. The first four moments of random variables should be sufficient, but it requires more computational effort when dealing with large number of samples. This method is recently used in probability analytical target cascading [40] and non-linear computational fluid dynamics calculations [41]. The Most Probable Point method was firstly developed in the reliability analysis [42]. It is then formally defined in a coordinate system of independent and standardized normal vector. A symmetrical joint probability distribution function would be centered at the origin in the output space. The most probable point is the point that contributes most to the probability estimation integral of the shortest distance from the origin to a point on the limit state surface in the output space. This method is used in Pratt and Whitney (PW) Engine Design [42] and fatigue reliability analysis [43].

The approach we propose here for propagating uncertainty through a computational

model involves some aspects of sample based methods and the development of a data-fit surrogate model via projection of the data onto a functional basis representation. To ensure scalability of our approach with increasing input dimension, we rely on an assumption of approximate additivity of the underlying full computational model. To introduce this concept, we consider first the high-dimensional model representation (HDMR) of a function, $f(x)$, which is written as [44, 45, 46],

$$f(x_1, x_2, ..., x_d) = f_0 + \sum_{j=1}^{d} f_j(x_j) + \sum_{j<l}^{d} f_{j,i}(x_j, x_l) + ... + f_{1,2,...,d}(x_1, x_2, ..., x_d) = \sum_{\boldsymbol{u} \subseteq \mathcal{D}} f_{\boldsymbol{u}}(\boldsymbol{X_u}), \tag{1.2}$$

where $\mathcal{D} := 1, 2, ..., d$ denotes the set of input indices, $\boldsymbol{u}$ is a multi-index, and individual terms in each summand are referred to as subfunctions. For square-integrable functions, which is the class of functions we are concerning ourselves with in this paper (i.e., those functions with finite variance when the inputs are random variables), we can write the functions variance as

$$\mathbb{V}(f) = \sum_{\substack{\boldsymbol{u} \subseteq \mathcal{D} \\ \boldsymbol{u} \neq \emptyset}} \mathbb{V}(f_{\boldsymbol{u}}), \tag{1.3}$$

which is a sum each individual subfunctions variance. In variance-based global sensitivity analysis, sensitivity indices for different combinations of inputs are estimated by normalizing each individual subfunctions variance by the total variance of f [47]. We make use of this concept in our greedy approach to determining which subfunctions to include in our surrogate representation as discussed in the chapter 2. Given the variance of a function as written above, we can define the effective superposition dimension of the function. Following reference [48], the effective superposition dimension is the smallest integer, d_s, such that

$$\sum_{\substack{\boldsymbol{u}\leq d_s \\ \boldsymbol{u}\neq\emptyset}} \mathbb{V}(f_{\boldsymbol{u}}) \geq \alpha \mathbb{V}(f), \tag{1.4}$$

where α is a user defined constant, such as 0.99. If a function has a low effective superposition dimension, then higher order interactions are not important in the construction of the HDMR in equation 1.2. It turns out, that a great deal of functions, particularly in the finance community, but also in engineering, have low effective dimension [11, 12, 48, 49, 50].

For functions with low effective dimension, e.g., $d_s = 2$, then the function is approximately additive in the sense that it consists primarily of subfunctions that are a function of only one input, with a few potentially significant subfunctions of more than one input. We seek to exploit the approximate additivity of many computational models by using the tools of compressed sensing to interrogate the low order subfunctions of a given function, f, and determine if the function is well-represented as approximately additive. If so, then uncertainty propagation can be carried out with a surrogate model constructed by summing the low-order subfunctions found by compressed sensing, which, as shown in chapter 3, results in substantial computational savings as compared to traditional Monte Carlo methods with nearly zero loss of accuracy.

Compressed sensing is a recently developed technique for recovering sparse signals that relies on linear dimensionality reduction [51, 52, 53]. We provide a brief overview of the rich field here. The general concept, as we employ it here, is that certain signals (for us outputs of computational models) can be approximated well by a sparse representation in a particular functional basis. Thus, the coefficient vector in the functional basis requires only a few nonzero entries. For a given set of basis functions, $\{\Psi_k\}_{k=1}^N$, we assume that a signal, f, can be represented as a linear combination

$$f = \Psi c, \tag{1.5}$$

where Ψ is an $N * N$ matrix with columns, Ψ_k, and $\boldsymbol{c}$ is an $N * 1$ vector of coefficients. If f is sparse (or approximately sparse) in the basis, Ψ, then $\boldsymbol{c}$ will consist of many values that are effectively zero. The function is called S-sparse in Ψ if there exists a $c \in \mathbb{R}^N$ with only $S \ll N$ nonzero entries. Samples of the signal, f, are obtained by another linear operator, Φ, which is an $M * N$ measurement matrix, where $M < N$. A requirement of compressed sensing is that Φ and Ψ be as incoherent as possible, meaning as dissimilar as possible [53]. We accomplish this here by setting Φ equal to a random subset of the rows of the identity matrix. Then the sampled signal is

$$b = \Phi f, \tag{1.6}$$

which in our context, is just M (rather than N, or even n in the context of Monte Carlo simulation with equation 1.1) evaluations of our computational model. The purpose of compressed sensing is then to recover the sparsest signal, $\Psi\boldsymbol{c}$, that produces the measurements f. This can be written as an optimization problem as

$$\hat{\boldsymbol{c}} = \arg \min_{\boldsymbol{c} \in \mathbb{R}^N} \|\boldsymbol{c}\|_0 \text{ subject to } b = \Phi\Psi\boldsymbol{c}, \tag{1.7}$$

where $\|\boldsymbol{c}\|_0$ is defined as the number of nonzero entries in $\boldsymbol{c}$. Finding a solution to this problem would require enumeration of all possibilities and is thus of combinatorial complexity. The fundamental insight in compressed sensing is the convex relaxation of equation 1.7 by using the l_1 norm to find the coefficients as

$$\hat{\boldsymbol{c}} = \arg \min_{\boldsymbol{c} \in \mathbb{R}^N} \|\boldsymbol{c}\|_1 \text{ subject to } b = \Phi\Psi\boldsymbol{c}, \tag{1.8}$$

where $\|\boldsymbol{c}\|_1 = \sum_{k=1}^{N} |\boldsymbol{c}_k|$. With enough measurements, if f is sparse in Ψ, then it can nearly always be reconstructed from $\boldsymbol{b}$ using equation 1.8, as $f \approx \Psi\hat{\boldsymbol{c}}$ [54]. equation 1.7 can be implemented as a linear program, for which many efficient solution algorithms exist. This is proved by [54], if f is a superposition of $|T|$ spikes $f(t) = \sum_{\tau \in T} f(\tau)\delta(t-\tau)$ obeying

$$|T| \leq C_M \cdot (logN)^{-1} \cdot |\Omega|\,, \tag{1.9}$$

where C_M is some contant, which is bigger than zero, f is a discrete-time signal, and Ω is a randomly chosen set of frequencies. With probability bigger than $1 - O(N^{-M})$, f could be reconstructed as the solution to the l_1 minimization statement

$$\min_{g} \sum_{t=0}^{N-1} \left|g(t)\right|, \text{ s.t. } \hat{g}(\omega) = \hat{f}(\omega) \text{ for all } w \in \Omega. \tag{1.10}$$

Our goal then in this work is to assume a basis exists such that the subfunctions of a given function, f, can be represented sparsely in that basis. By assuming low effective superposition dimension, we will then interrogate the subfunctions of f by fixing all inputs that are not active in a given subfunction. Doing this allows us to sparsely reconstruct the subfunctions of f with very few evaluations of f per subfunction. If the function is of low effective superposition dimension, then only a small number of subfunctions must be reconstructed and then added together to provide a nearly exact surrogate representation of f. We develop our methodology for constructing such surrogates in the following section.

1.4 Thesis organization

My thesis is organized as follows: the 1st chapter presents the introduction and background, the 2nd chapter presents the approach that used in this thesis, the 3rd chapter presents the application of the approach and its results, and the 4th chapter presents the summary and conclusions of this thesis.

2. APPROACH *

In this work we assume we have a sparse representation of a given function f in the basis of Legendre polynomials or Fourier series. Legendre polynomials basis and Fourier basis are first introduced. The proposed approach in this thesis is presented, followed by some possible methods to build computational models and some possible methods to eliminate unreasonable computational models.

2.1 Legendre polynomials basis

The Legendre polynomials in one-dimension, L_n, on $[-1, 1]$, can be written as

$$L_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n}[(x^2 - 1)^n], \tag{2.1}$$

according to Rodrigues formula. These polynomials are orthonormal with respect to the uniform measure on $[-1, 1]$. When attempting find a sparse representation of a function, f, using equation 1.8, we randomly sample a few points in the input space to generate a few random samples of the output space (this ensures good incoherence of Ψ and Φ). For the case of Legendre polynomials, sampling randomly from the Chebyshev distribution, rather than uniformly, produces better recovery of signal information [55, 56]. Thus, to construct Ψ, we sample in the input space with the Chebyshev distribution (also known as the arcsine distribution), whose probability density function is

$$p(x) = \frac{1}{\pi\sqrt{1 - x^2}}, \tag{2.2}$$

*Reprinted with permission from "A compressed sensing approach to uncertainty propagation for approximately additive functions" by Kaiyu Li, Douglas Allaire, 2016. ASME 2016 International Design Engineering Technical Conferences, IDETC/CIE, Copyright 2016 by ASME.

with support $[-1, 1]$. To ensure we can handle arbitrary intervals, $[a, b]$, with the Legendre polynomials, we sample a Chebyshev point and bijectively map it onto $[a, b]$ as

$$x \mapsto a + (b - a)\frac{x + 1}{2}. \tag{2.3}$$

To construct a Legendre polynomial basis for a function of d inputs, with order 0 to $N_j - 1$ one-dimensional Legendre polynomials in each dimension, j, we take tensor products. Let $\mathcal{L}_j$ be a matrix whose rows are made up of the Legendre polynomials evaluated at given samples of x_j. That is, each row has a specific sample of x_j associated with it. If we have M_j samples and order 0 through $N_j - 1$ Legendre polynomials for input j, then $\mathcal{L}_j$ is $M_j \times N_j$. We can then construct the matrix Ψ as

$$\Psi = \bigotimes_{j=1}^{d} \mathcal{L}_j, \tag{2.4}$$

where $\bigotimes$ is the Kronecker product of the matrices, $\mathcal{L}_j$, and Ψ is $M \times N$, where $M = \prod_{j=1}^{d} M_j$ and $N = \prod_{j=1}^{d} N_j$. If the tensor product is too complicated, we could use tensor-train decomposition method to solve [57].

2.2 Fourier basis

It is apparent that some functions could be represented simpler in Fourier series basis rather than Legendre polynomial basis, such as: sines and cosines functions. So Fourier series basis is used to compare with Legendre polynomial basis in some examples mentioned in the 3rd chapter. A Fourier series is defined as an expansion of a function or representation of a function in a series of sines and cosines [58]. It could be written as:

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos nx + \sum_{n=1}^{\infty} b_n \sin nx, \tag{2.5}$$

where

$$a_n = \frac{1}{\pi}\int_0^{2\pi} f(x)\cos nx dx, \tag{2.6}$$

$$b_n = \frac{1}{\pi}\int_0^{2\pi} f(x)\sin nx dx, n = 0, 1, 2, ..., \tag{2.7}$$

To keep all the other influencing factors the same, the overall sampling process is the same as the Legendre polynomials basis sampling process. Some points in the input space are randomly sampled, and the output is generated. The sampling of the input space comes from the Chebyshev distribution, the same bijectively map was used as mentioned in equation 2.3. To construct a Fourier series basis for a function with N_j degrees in each input dimension j, we would have the following basis constructed:

$$[1, \cos x_j, \sin x_j, \cos 2x_j, \sin 2x_j, \cos 3x_j, \sin 3x_j, ..., \cos \lceil\frac{N_j}{2}\rceil, \sin \lceil\frac{N_j}{2}\rceil], \tag{2.8}$$

where x_j is the input of dimension j. The tensor product is calculated the same way as the Legendre polynomials basis.

2.3 Surrogate representation

As can be seen by the formula for M, the number of samples required grows combinatorially with the number of inputs. Thus, even just a few samples in each dimension can quickly become computationally infeasible if d is large and we are creating a basis over the product space of the inputs using equation 2.4. However, if we assume f is approximately additive, then there is no need to create a basis over the product space of the inputs, since higher order interaction terms in the HDMR of the function will be negligible. For example, if we know that the superposition effective dimension of a function, f, is $d_s = 1$, then the function (which is then referred to as purely additive) can be written exactly as

$$f(\boldsymbol{x}) = f_0 + \sum_{j=1}^{d} f_j(x_j). \tag{2.9}$$

To construct a surrogate representation of an f defined by equation 2.9, we need only construct individual surrogates for each f_j. To do this in our compressed sensing context, we can define $\Psi_j = \mathcal{L}_j$ and solve equation 1.8 to arrive at a sparse surrogate representation,

$$\hat{f}_j \approx f_j + \hat{f}_0, \tag{2.10}$$

of the subfunction associated with x_j and an offset, $\hat{f}_0$. The surrogate estimates an offset because we must fix the inputs $\{x_l\}_{l \neq j}$ to nominal values. We use $\hat{f}_0$ here rather than f_0 or $\hat{f}_0$ because the offset in this expression depends entirely on the nominal values (so it is not unique in the sense of having only one possible f_0) of the inputs chosen and it is not an approximation. This concept is shown notionally for a two input function in the left plot of figure 2.1, where the squares represent samples of x_1 with x_2 fixed to a nominal value to estimate f_1, and the circles represent samples of x_2 with x_1 fixed to a nominal value. In each dimension, our approach is to start with a small number of samples (e.g., 3), and solve equation 1.8. We then compute $\left\|\hat{c}_j\right\|_0$. We then randomly add another Chebyshev point, solve equation 1.8 , and compute $\left\|\hat{c}_j\right\|_0$ again. Once this l_0 norm reaches equilibrium, we assume we have arrived at a good representation of f_j. This is a heuristic for determining how many samples are required for any given input and requires further study. Certainly if $\left\|\hat{c}_j\right\|_0 = N_j$, then we do not have sparsity in the basis representation and the representation itself is likely a poor approximation of the subfunction f_j plus an offset. If this conclusion is reached at this point, then at worst we have spent $\sum_{j=1}^{d} N_j$ function evaluations, which can be repurposed for another approximation approach based on random sampling or used as part of a Monte Carlo simulation. If our sampling heuristic in each dimension of an additive function suggests we have a good sparse representation of each subfunction, then

we may construct our surrogate of f as

$$\hat{f}(\boldsymbol{x}) = \tilde{f}_0 + \sum_{j=1}^{d} \hat{f}_j - d\tilde{f}_0, \tag{2.11}$$

where the offset term, $\tilde{f}_0$ is computed by evaluating f at a nominal value of each input.

2.4 Subfunction approximation

To determine how well our additive surrogate representation is approximating f, we conduct a validation exercise after each additional term in the surrogate HDMR is added. we would follow a method called cross validation holdout method, or called test sample estimation [59], which divide the data set into two mutually exclusive subsets called training set and test set. The training set is used to generate the model, and the test set is used to evaluate the accuracy of the model. So we randomly sample a small number of points from $\boldsymbol{x}$, (e.g., 10 points), and evaluate f for each of these points. None of these points is a part of the point set used to generate the surrogate representation. At these points we compute the difference between our surrogate representation and the true function, $e_i(\boldsymbol{x}^i) = f(\boldsymbol{x}^i) - \hat{f}(\boldsymbol{x}^i)$. We then use $\|e\|_2$, as an indicator of how well the surrogate is approximating f. As a heuristic, if $\|e\|_2$ is below a user specified threshold, we consider the approximation sufficient. The implications of this heuristic are a topic of future work.

If, after adding all subfunctions of one input, that is, all possible $\hat{f}_j$ terms, we are still not approximating f well in our validation set, then we continue on to two input subfunctions. This is assuming also that a sparse representation for each subfunction, f_j, has been found. Moving to subfunctions of two inputs requires the tensor product of the Ψ_j matrices. For example, for a subfunction approximation of $f_{j,l}$, we compute the matrix $\Psi_{j,l} = \Psi_j \bigotimes \Psi_l$. This requires a tensor product of the onedimensional Chebyshev point sets for x_j and x_l , and subsequent evaluation of f at those points. A two-dimensional tensor product of 9 Chebyshev points in each dimension is shown in the right plot of

figure 2.1.

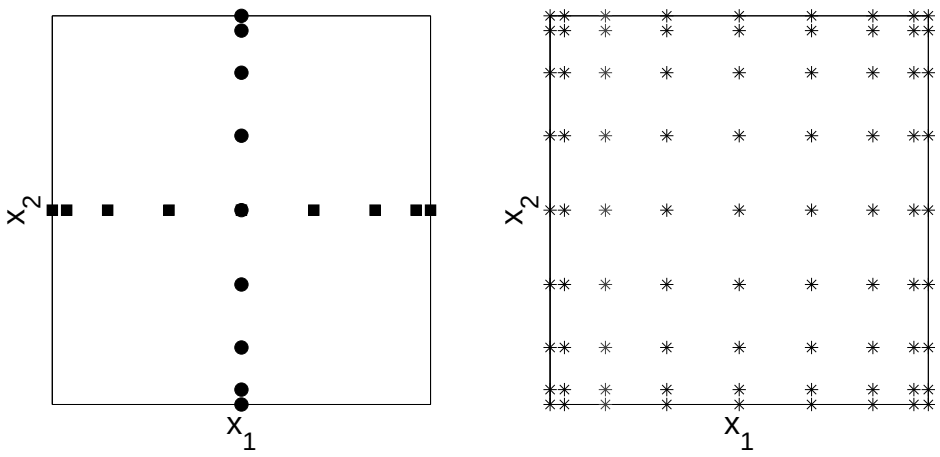

Figure 2.1: One dimensional Chebyshev distributed points for two inputs for estimating one input subfunctions (left). Tensor product of the one dimensional Chebyshev distributed points for estimating a two input subfunctions (right).

Here an original set of 18 points used to estimate two single input subfunctions would become 81 points. Thus, to ensure that only two-input subfunctions of likely significance in the approximation of f are incorporated, we use a greedy-based procedure for selecting which interaction terms to use. For this, we estimate the main effect sensitivity indices for each input x_j as

$$\hat{S}_j = \frac{\mathbb{V}(\hat{f}_j)}{\mathbb{V}(\hat{f})}. \tag{2.12}$$

These sensitivity indices estimate the approximate main effect contribution of input, x_j, and should be estimated using the anticipated distributions of each input that will be used in uncertainty propagation through the surrogate once its construction is complete. Experience has shown that inputs with large main effect indices also tend to be those inputs

involved in significant interaction effects.

However, when the nonlinearity part and the interaction part among variables become a considerable fraction of the output variance, the input variables with larger main effect does not necessarily means there is large interaction effect between those inputs. For example, in the problem mentioned in [60], a second-order polynomial function is being considered.

$$f(x) = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_{11} x_1^2 + \beta_{12} x_1 x_2, \tag{2.13}$$

where $\beta_1 = \beta_2 = \beta_3 = 1/100$, $\beta_4 = 1/10$, $\beta_{11} = 1$, and $\beta_{12} = 1$, and all the input variables are uniformly distributed in $[-1, 1]$. And the sensitivity of the input variables are:

$$\begin{aligned} S_1 = V_1/V = 0.4371, S_2 = S_3 = V_2/V = 1.639e-4, \\ S_4 = V_4/V = 1.639e-2, S_{12} = V_{12}/V = 0.5462. \end{aligned} \tag{2.14}$$

It is obvious to see that the input β_1 and β_4 has the largest main sensitivity, but the interaction effect between input β_1 and β_2 has the largest sensitivity.

As input variables has a higher main effect usually means those input variables have a higher interactive effect, we could still add two input subfunctions by starting with the input with the largest sensitivity index and the input with the second largest sensitivity index. If our heuristic error indicator has not fallen below the threshold after this subfunction is incorporated into the surrogate approximation, then we continue with the two input subfunction of the inputs with the largest and third largest sensitivity index, and continue in this manner until we have satisfied our error threshold, or determined it was wise to abandon the approximation in this particular basis representation. We note here that when two

input subfunctions have been added to the surrogate representation, we must take care to avoid double-counting single input subfunction terms. That is, our two input subfunction approximation for inputs x_j and x_l is:

$$\hat{f}_{j,l}(x_j, x_l) \approx f_{j,l}(x_j, x_l) + f_j(x_j) + f_l(x_l) + \hat{f}_0. \tag{2.15}$$

Thus, if all possible two input subfunctions are added to the surrogate representation of f, when $d \geq 3$, we have

$$\hat{f}(\boldsymbol{x}) = \frac{(d-1)(d-2)}{2}\tilde{f}_0 - (d-2)\sum_{j=1}^{d}\hat{f}_j(x_j) + \sum_{\substack{j,l=1 \\ j<l}}^{d}\hat{f}_{j,l}(x_j, x_l). \tag{2.16}$$

Once we have a surrogate representation of $f(\boldsymbol{x})$ that we are satisfied with, we can proceed to propagate uncertainty through the surrogate using any supported distribution of $\boldsymbol{x}$. That is, any distribution whose probability density has values greater than zero only on the hypercube defined by the maps for each input defined by equation 2.3 and used in the surrogate construction. We note that our approach also provides approximations to the main effect sensitivity indices of each input, and variability of these sensitivities with respect to changing input distributions can rapidly be assessed.

However, this method does not work very well when the functions are not additive. Take Breguet range equation [61] as an example,

$$Range = V \times (\frac{L}{D}) \times I_{sp} \times \frac{W_i}{W_f}. \tag{2.17}$$

This function has four input variables: $V, L/D, I_{sp},$ and W_i/W_f. If the Legendre polynomials degree is 50, then the tensor product matrix size is $50 \times 50 \times 50 \times 50$, which is too large and significantly decreases the computational efficiency.

2.5 Gaussian process

A Gaussian process is a collection of random variables and any finite number of random variables would have joint Gaussian distribution [62]. Gaussian process method provides a practical and probabilistic approach to learning in kernel machines. Gaussian process measures the similarity between the points and it gives a prediction unseen points value. This method not only gives the prediction estimation of the point, but also gives the uncertainty information (marginal distribution). Gaussian process is a stochastic process specified by its mean function and covariance function. These functions contain the information of functional form, and consist a set of parameters called hyperparameters. The mean function is expressed as:

$$\mu(\boldsymbol{x}) = \mathbb{E}[(f(\boldsymbol{x}))]. \tag{2.18}$$

The covariance function is expressed as:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}[(f(\boldsymbol{x}) - \mu(\boldsymbol{x}))(f(\boldsymbol{x}') - \mu(\boldsymbol{x}'))]. \tag{2.19}$$

Given a dataset, $\mathcal{D} = \{(x_i, y_i)|i = 1, 2, ..., n\}$, Gaussian process regression purpose is to predict the function value y_* at x_*, and the actual function value is denoted as f_*. The covariance function among all possible combinations of all the points in the dataset is denoted as k and some variables are defined for convenience:

$$K = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & k(x_1, x_3) & \dots & k(x_1, x_n) \\ k(x_2, x_1) & k(x_2, x_2) & k(x_2, x_3) & \dots & k(x_2, x_n) \\ \dots & \dots & \dots & \dots & \dots \\ k(x_n, x_1) & k(x_n, x_2) & k(x_n, x_3) & \dots & k(x_n, x_n) \end{bmatrix}, \tag{2.20}$$

$$K_* = \begin{bmatrix} k(x_*, x_1) & k(x_*, x_2) & k(x_*, x_3) & \dots & k(x_*, x_n) \end{bmatrix}, \tag{2.21}$$

$$K_{**} = k(x_*, x_*). \tag{2.22}$$

The key assumption in Gaussian process modeling is that the data could be represented as a sample from Gaussian distribution, so we could have the following equation:

$$\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix} \sim \mathcal{N}(0, \begin{bmatrix} K & K_*^T \\ K_* & K_{**} \end{bmatrix}), \tag{2.23}$$

It is proved that [63] the following probability is a Gaussian distribution:

$$y_* | \boldsymbol{y} \sim \mathcal{N}(K_* K^{-1} \boldsymbol{y}, K_{**} - K_* K^{-1} K_*^T). \tag{2.24}$$

The mean of y_* is:

$$\overline{y_*} = K_* K^{-1} \boldsymbol{y}, \tag{2.25}$$

and the variance of y_* is:

$$\text{var}(y_*) = K_{**} - K_* K^{-1} K_*^T. \tag{2.26}$$

Although Gaussian process could be applied to infinite dimensional objects, the prediction complexity of finite dataset is $O(n^3)$.

The GPML Matlab code is used in this paper to predict the function with limited input points. The Gaussian likelihood function is used as the likelihood function. The inference method used here is exact interence for GP with Gaussian likelihood. This method calculates a parametrization of the posterior, the negative log marginal likelihood and its derivatives with respect to the hyperparameters. The results are being compared with

the compressed sensing method using Legendre polynomials basis or Fourier series basis. From the uncertainty information from the GP method, it is obvious that there exist many possible functions given the input points and output values. And we want to construct a way of display such uncertainty information similar as GP method for compressed sensing approach. Thus, some possible methods are used to explore those uncertainty information.

2.6 Brownian Bridge

Brownian motion [64] or Wiener process is defined as a sample-continuous Gaussian Process x on $T = [0, \infty]$ with mean 0 and covariance as:

$$E_{x_s,x_t} = \min(s, t). \tag{2.27}$$

Brownian Bridge is closely related to Brownian motion. Brownian Bridge is defined as a sample-continuous Gaussian Process y on $T = [0, 1]$ with mean as 0 and covariance as:

$$E_{y_s,y_t} = s(1 - t) \text{ for } 0 \leq s \leq t \leq 1. \tag{2.28}$$

The expected value of Brownian Bridge is zero, and the variance at $T = 0$ and $T = 1$ is zero, so it means $y_0 = y_1 = 0$. As a result, the starting node and the ending node of the Brownian Bridge is at the same level. Given a Brownian motion x, a Brownian Bridge y could be obtained with the following equation:

$$y_t := x_t - tx_1, 0 \leq t \leq 1, \tag{2.29}$$

where x_1 stands for the Brownian motion value (x) when $t = 1$. In a more general setting, a Brownian Bridge connecting the points $(0, a)$ to (T, b) could be defined as [65]:

$$y_t^{a\to b} := a(1 - \frac{t}{T}) + b\frac{t}{T} + (x_t - \frac{t}{T}x_T), 0 \leq t \leq T, \tag{2.30}$$

where $y_t^{a \to b}$ is a Brownian Bridge from point $(0, a)$ to point (T, b), and x_T stands for the Brownian motion value (x) when $t = T$.

Random walk is a good substitute for Brownian Motion [66], thus it could be used to construct Brownian Motion. Let X_n is an independent variable, it only has two possible values: -1 and 1, with equal probability each, which means X_n has 0.5 probability equal to 1, and 0.5 probability equal to -1. Then S_n is defined as a random walk process with:

$$S_n := X_1 + X_2 + X_3 + ... + X_n. \tag{2.31}$$

Figure 2.2 is a collection of 1000 Brownian Bridges starting from point $(0, -5)$ and ending at point $(2, 5)$, using random walk process as a build for Brownian Bridges.

But in our following sections, we decide to use uniform random walk process [67] as a method of building Brownian Bridge. This is because we could get more variability from Brownian Bridges generation results. Currently, we are having many possible lines going through the starting point and the ending point, and some uncertainty information for those possible functions could be obtained. But apparently, not all the possible lines could be considered reasonable, as some functions are really different from the real function. For those functions that have a huge difference from the real functions, some possible approaches are being proposed to eliminate those functions. The first method we proposed is to use the modified Lipschitz function related filtering method, and the second method is to use modified VC-dimension related filtering method. After filtering, we expect to get a more accurate uncertainty information for our approach.

2.7 Modified Lipschitz filtering approach

Given two metric spaces (X, d) and (Y, e) [68], where d denotes the metric on the set X, and e denotes the metric on the set Y, which means: $f : X \to Y$. If there exists a real constant k, which satisfy:

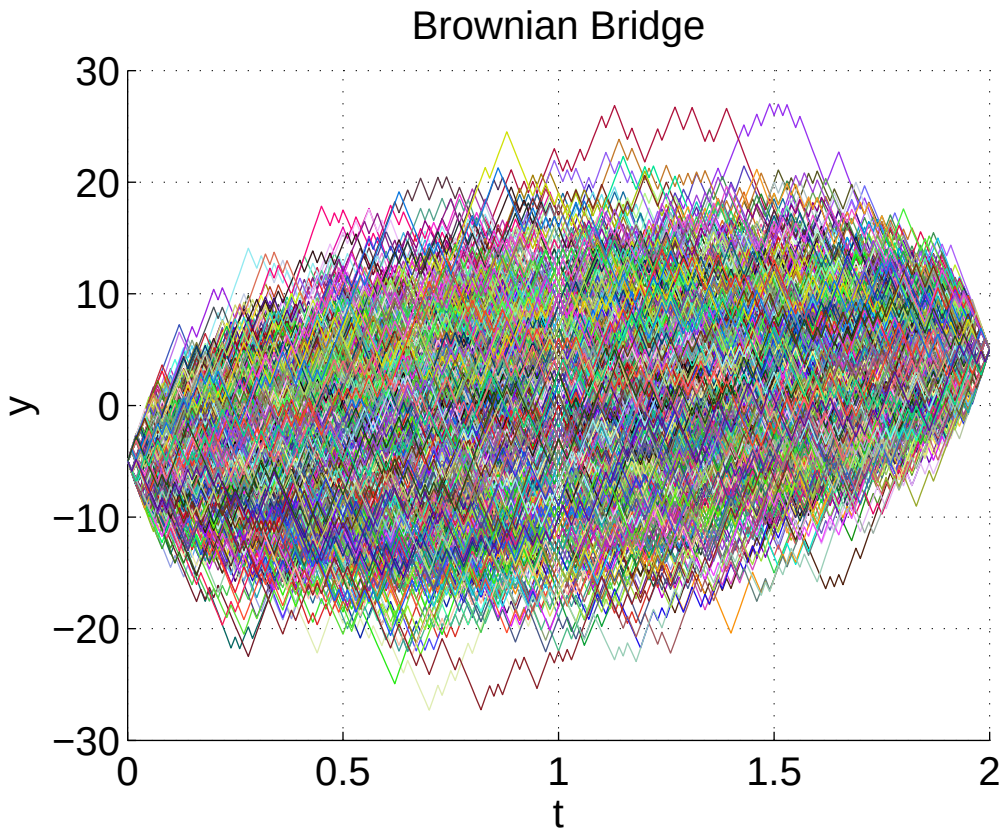

Figure 2.2: 1000 Brownian Bridges between point $(0, -5)$ and $(2, 5)$

$$e(f(a), f(b)) \leq kd(a, b), \tag{2.32}$$

for all $a, b \in X$, then f is defined as a Lipschitz function and the Lipschitz constant is defined as k. If S is a subset of X, then f is uniformly continuous on S. Obviously, the constant k here in Lipschitz function could be used as a method to eliminate some Brownian Bridges that do not satisfy some certain requirement.

Using equation 2.30, we try to construct a Brownian Bridge from two adjacent points of compressed sensing approach samples, which are: (x_s, y_s) and (x_e, y_e). Map the starting point of this Brownian Bridge as $(0, a)$ and the ending point of this Brownian Bridge as

(T, b), where $a = y_s$, $T = t_e - t_s$, and $b = y_e$. And we try to calculate the slope between two random adjacent points from a random selected Brownian Bridge, which are: (t_0, y_0) and (t_1, y_1). Then the slope k_{BB} between the two adjacent points on the Brownian Bridge is:

$$k_{BB} = \frac{1}{t_1 - t_0}(\frac{b-a}{T} + (x_{t_1} - x_{t_0}) - \frac{x_T}{T}), \tag{2.33}$$

where x_t definition is the same as it is in equation 2.30. It is obvious to see that k_{BB} is full related with the slope between the Brownian Bridge starting point and the Brownian Bridge ending point as it is related to a, b and T. So, we propose a possible approach to decrease the influence of this factor. We could calculate the angle between this part of randomly generated Brownian Bridge (from (t_0, y_0) to (t_1, y_1)) and the line connecting the starting point (x_s, y_s) and the ending point (x_e, y_e) of Brownian Bridge. If this angle is within some reasonable range, we conclude that this part of Brownian Bridge satisfies our selection requirement. If all the parts on this Brownian Bridge have angle within the reasonable range, we conclude that this Brownian Bridge satisfies our selection requirement.

2.8 Modified VC-dimension filtering approach

In this section, we would like to propose another possible method to help us build better uncertainty information for our approach, which is to build bounds on the risk for loss functions. The concept of VC-dimension is briefly introduced here first.

Given a set system (X, F) and a subset $Y \subseteq X$ [69], the restriction of F on Y is the set $F|_Y = S \cap Y : S \in F$. There exist a subset $A \subseteq X$ is defined as shattered by F, if for some $S \in F$, each of the subsets of A arises as an interaction $A \cap S$, which could be denoted as $F|_Y = 2^A$. The VC-dimension $dim(F)$ of F is defined as the size of the largest subset of X which could be shattered by F. Take the following sets as an example. Considering X as the Euclidean plane $\mathbb{R}^2$, F_1 as the set all the possible convex polygons

pn the plane, and F_2 as the set of all the possible halfplanes. The set system (X, F_1) would have a VC-dimension of ∞, and the set system (X, F_2) would have a VC-dimension of 3. Given $R[x_1, x_2, ..., x_d]_1$ as the set of all the possible linear functions in d variables, the set $P_{d,1}$ denotes:

$$P_{d,1} = \{\{x \in \mathbb{R}^d : p(x) \geq 0\} : p \in R[x_1, x_2, ..., x_d]_1\}. \tag{2.34}$$

For the set system $(\mathbb{R}^d, P_{d,1})$, the VC-dimension is $d + 1$. More generally, given $R[x_1, x_2, ..., x_d]_{\leq D}$ as the set of all the possible real polynomials functions in d variables of degree at most D, the set $P_{d,D}$ denotes:

$$P_{d,D} = \{\{x \in \mathbb{R}^d : p(x) \geq 0\} : p \in R[x_1, x_2, ..., x_d]_{\leq D}\}. \tag{2.35}$$

For the set system $(\mathbb{R}^d, P_{d,D})$, the VC-dimension is at most $\binom{D+d}{d}$.

After the brief introduction of the concept of VC-dimension, we would like to first introduce methods to build bounds on the risk for simple loss functions, like indicator loss functions. Vapnik [70] proposed an empirical method for measuring the capacity of a learning machine, given the size of training set l, with the input-output pairs (x, ω), $x \in X \subset R^n, \omega \in \{0, 1\}$, $(x_1, \omega_1), ..., (x_l, \omega_l)$, where all the pairs are assumed to be drawn independently from an unknown distribution. The learning machine is characterized by the set of binary classification functions $f(x, \alpha)$, and $\alpha \in \Lambda$, where α is a parameter that specifies the function, Λ is the set of all admissible parameters, the loss function between the value ω and the output value from the learning machine $f(x, \alpha)$ is defined as $p(\alpha)$:

$$p(\alpha) = E|\omega - f(x, \alpha)|. \tag{2.36}$$

The empirical error $v(\alpha)$ is defined as:

$$v(\alpha) = \frac{1}{l}\sum_{i=1}^{l}\left|\omega_i - f(x_i, \alpha)\right|. \tag{2.37}$$

Vapnik shows that we could establish an upper bound for $p(\alpha)$ based on three aspects of knowledge, which are: the capacity of learning machine (measured by VC-dimension), the empirical risk $v(\alpha)$, and the size of training set l. With probability $1-\eta$, for all $\alpha \in \Lambda$

$$p(\alpha) \leq v(\alpha) + D[l, h, v(\alpha), \eta] \tag{2.38}$$

is valid, where D could be expressed as:

$$D[l, h, v(\alpha), \eta] = c\frac{h[\ln 2l/h + 1] - \ln \eta}{2l}\{\sqrt{1 + \frac{4lv(\alpha)}{c[h[\ln 2l/h + 1] - \ln \eta]}} + 1\}, \tag{2.39}$$

where c is defined as a universal constant, less than 1, and h is the VC-dimension.

But it is apparent that not all the function values could fit in the range set $\{0, 1\}$, those real-valued functions would have a more general bound on the risk for loss functions. In 1998, Vapnik [71]defines the loss function $Q(x, \alpha)$ as:

$$Q(x, \alpha) = (y - f(x, \alpha))^2, \tag{2.40}$$

where y is the real function value from the training set (x_i, y_i), and $f(x, \alpha)$ is the output of the approximate function. If we assume the loss function is a set of totally bounded functions, which means ($0 \leq Q(x) \leq B$), then with the probability at least $1-\eta$ the bound

$$R(\alpha) < R_{emp}(\alpha) + \frac{B\varepsilon(l)}{2}(1 + \sqrt{1 + \frac{4R_{emp}(\alpha)}{B\varepsilon(l)}}) \tag{2.41}$$

is valid, where $R(\alpha)$ is the risk function:

$$R(\alpha) = \int Q(x, \alpha)dx, \alpha \in \Lambda, \tag{2.42}$$

$R_{emp}(\alpha)$ is the empirical risk function:

$$R_{emp}(\alpha) = \frac{1}{l}\sum_{i=1}^{l} Q(x, \alpha), \alpha \in \Lambda, \tag{2.43}$$

where α is a parameter that specifies the function, Λ is the set of all admissible parameters, l is the size of training set, and $\varepsilon(l)$ could be expressed as:

$$\varepsilon(l) = 4\frac{h(\ln(2l/h) + 1) - \ln(\eta/4)}{l}, \tag{2.44}$$

where h is the VC-dimension. In our problem, we would assume within range $[x1, x2]$, the real function $f(x)$ would be within the range: $[R_l, R_u]$, and the result from compressed sensing approach is within the range: $[CS_l, CS_u]$. Thus, it is safe to conclude that the B in equation 2.41 would satisfy:

$$B \leq (\max(CS_u, R_u) - \min(CS_l, R_l))^2. \tag{2.45}$$

We would propose a preliminary method to find the appropriate value for B. We would decrease the B starting from $(\max(CS_u, R_u) - \min(CS_l, R_l))^2$, and ending at a number where we determine that this modified VC-dimension approach would give us a reasonable uncertainty information for our compressed sensing approach. Thus, we would use equation 2.41 as our main equation in eliminating unreasonable Brownian Bridges. The first step is to calculate the right side of equation 2.41, which is an upper bound distance measurement (expressed as d_1) between the real function and the function result from compressed sensing method. Then we would calculate the left side of equation 2.41

based on the distance measurement (expressed as d_2) between possible Brownian Bridges and function result from compressed sensing method in aspect of fixed input intervals. If for one Brownian Bridge, we find that $d_2 > d_1$, it is safe to conclude that this Brownian Bridge is not a possible substitute function for the real function using this modified VC-dimension related method, and this Brownian Bridge is filtered.

3. APPLICATION AND RESULTS *

We demonstrate our compressed sensing approach for uncertainty propagation on a one dimensional simple Legendre basis function, one dimensional complicated Legendre basis function, one dimensional simple Fourier basis function, two dimensional Legendre basis function, and a finite element model of a cooled gas turbine blade. The possible Brownian Bridges are built based on the sample points from compressed sensing approach. All the analytical functions are designed to show the effectiveness of our approach under perfect conditions, with comparison to Gaussian process method. The cooled gas turbine blade model was treated as a black box, and thus, nothing was known about the sparsity of this model in the Legendre basis. This is the type of scenario we expect to encounter generally. Finally, we use a simple example from compressed sensing approach with Legendre basis to demonstrate the process of using Brownian Bridges, modified Lipschitz filtering approach, and modified VC-dimension filtering approach to build uncertainty information for our method.

3.1 1-D simple Legendre function

The function that we applied in this section is:

$$f(x) = 1 + x + x^2, \tag{3.1}$$

where $x \in [-1, 1]$. This function could be expressed in Legendre polynomials in three terms, which could be expressed as the following way:

$$f(x) = \frac{4}{3} * 1 + 1 * x + \frac{2}{3} * \frac{1}{2}(3x^2 - 1), \tag{3.2}$$

*Reprinted with permission from "A compressed sensing approach to uncertainty propagation for approximately additive functions" by Kaiyu Li, Douglas Allaire, 2016. ASME 2016 International Design Engineering Technical Conferences, IDETC/CIE, Copyright 2016 by ASME.

where $1, x,$ and $\frac{1}{2}(3x^2 - 1)$ are the first three terms in Legendre polynomials. As we could use sparse Legendre basis to represent this function, it could be approximated perfectly with a few of evaluations and 5 samples are used in the compressed sensing approach. The reason of using 5 samples comes from the example of 2-D Legendre polynomials function in section 3.4. As 5 samples are enough to have a reasonable result for 2-D Legendre polynomials function, we would assume that 5 samples should be enough for the other relatively simpler functions. The estimated function and the real function are being shown in figure 3.1. The red line is the result from the compressed sensing method with Legendre basis and the black line is the real function. The blue dots are the samples randomly selected from compressed sensing method. In figure 3.2, the red line and the black line are the empirical cumulative distribution function (ECDF) line of the real function and the result from compressed sensing with Legendre basis separately. We want to use ECDF graph as a way of demonstration because we are trying to propogate input uncertainty x with uniform distribution on $[-1, 1]$ through the equation 3.1. It is obvious to see those two lines are basically the same, which means the result from compressed sensing approach with Legendre basis could give us the correct function.

However, if we try to use compressed sensing approach with Fourier basis with the same 5 samples as above, totally different result would be obtained. In figure 3.3, result from compressed sensing with Fourier basis and the real function are shown. In figure 3.4, the ECDF of result from compressed sensing method with Fourier basis and the real function are shown. As we could not use Fourier basis to express Legendre polynomials in limited terms (or in a sparse matrix), this method with Fourier basis does not work well on Legendre functions.

Using Gaussian process (GP) as a machine learning approach, the result from GP method and the real line are shown in figure 3.5. In figure 3.6, the ECDF of the result from GP method and the real function are shown. The mean function is composite, which

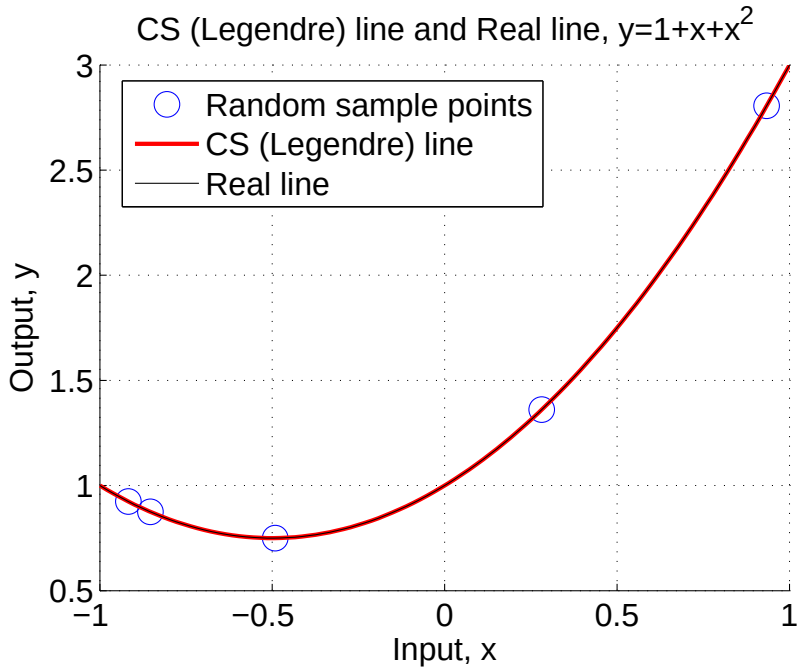

Figure 3.1: Compressed sensing approach with Legendre basis result and real function comparison on 1-D simple Legendre function.

is set as the sum of polynomial mean and constant mean. The covariance function is set as Matern form with isotropic distance measure. The hypermeters are minimized based on the sample points, the mean function and the covariance function. The likelihood function is set as Gaussian. Two of the five samples are set at the starting point and the ending point of the interval, which are: -1 and 1. The other three samples are randomly selected from the interval $[-1, 1]$. From the graph, we could use the GP method works as well as our method in this testing example.

3.2 1-D complicated Legendre function

The function that we applied in this section is more complicated than the previous section, which is:

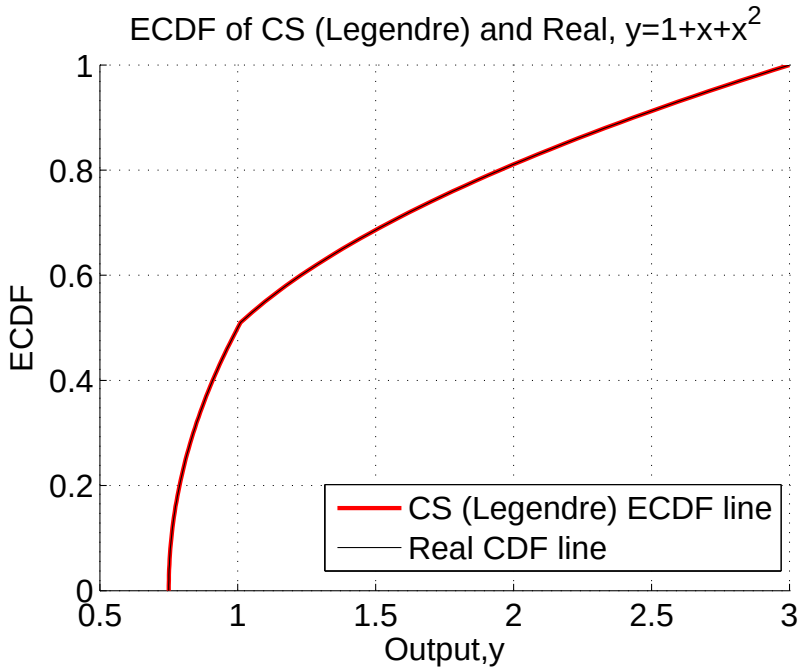

Figure 3.2: ECDF of compressed sensing approach with Legendre basis result and real function on 1-D simple Legendre function.

$$y = \frac{1}{128}(12155x^9 - 25740x^7 + 18018x^5 - 4620x^3 + 315x). \tag{3.3}$$

This function seems complicated, but it is the $9th$ term in Legendre polynomials, which means only one number is used to represent the function in Legendre basis. As we can see from the previous section, using compressing sensing with Fourier basis approach does not apply well on Legendre polynomials. As a result, only GP approach and compressed sensing approach with Legendre basis are considered in this section. In figure 3.7 and figure 3.8, we could see that compressed sensing with Legendre basis method work much better than the GP method in figure 3.9 and figure 3.10. As this function could be represented in a sparsely in Legendre basis, the compressed sensing approach with Legendre basis works great in this example. However, in GP approach, there are not enough samples for the

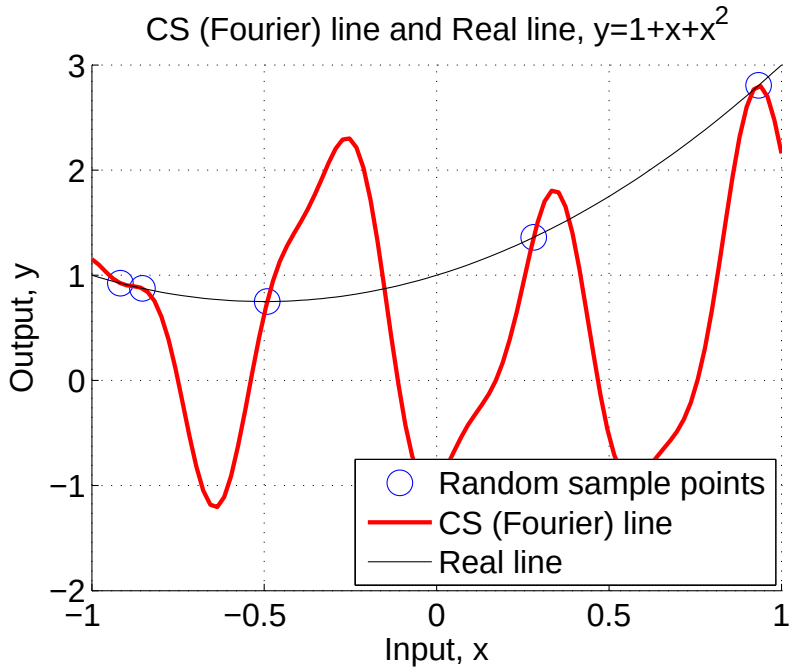

Figure 3.3: Compressed sensing approach with Fourier basis result and real function comparison on 1-D simple Legendre function.

approach to learn the function itself, and it could not obtain a perfect solution. For the samples used in GP approach, two of five samples are fixed at the interval starting point and ending point, and the other three samples are randomly selected.

With comparison among different Legendre polynomials functions using different methods, we could find that using compressing sensing approach with Legendre basis works better than Fourier basis while simulating Legendre polynomials functions. When the function could be expressed in Legendre polynomials in a sparse matrix, the compressed sensing approach is better than GP approach when the function is complicated.

3.3 1-D simple Fourier function

Apparently, not all the functions could be expressed sparsely in Legendre polynomials, for example, the Fourier series functions. However, simple Fourier function could be

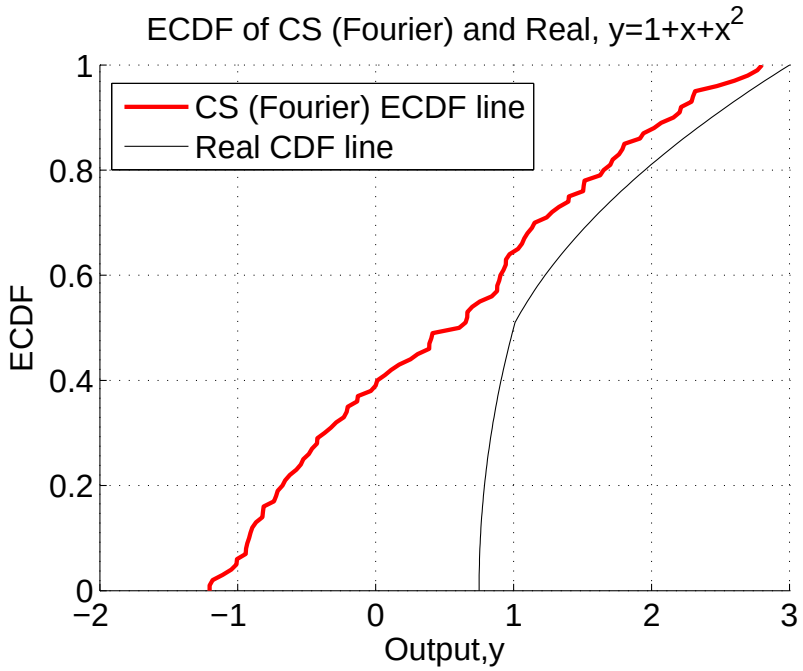

Figure 3.4: ECDF of compressed sensing approach with Fourier basis result and real function on 1-D simple Legendre function.

expressed sparsely with Fourier series basis. Compressed sensing method with Fourier series basis is applied into some Fourier function examples to see the effectiveness of our method. The function being applied in this section is:

$$y = \sin(15x). \tag{3.4}$$

This function could be expressed in Fourier series basis in a sparse matrix, which is only one non-zero number in the matrix. This problem is solved with compressed sensing approach with Fourier basis as shown in figure 3.11 and ECDF of the result and real function are shown in figure 3.12. This problem is also solved with compressed sensing approach with Legendre basis and GP method for comparison, as they are shown in figure 3.13, figure 3.14, figure 3.15, and figure 3.16 separately.

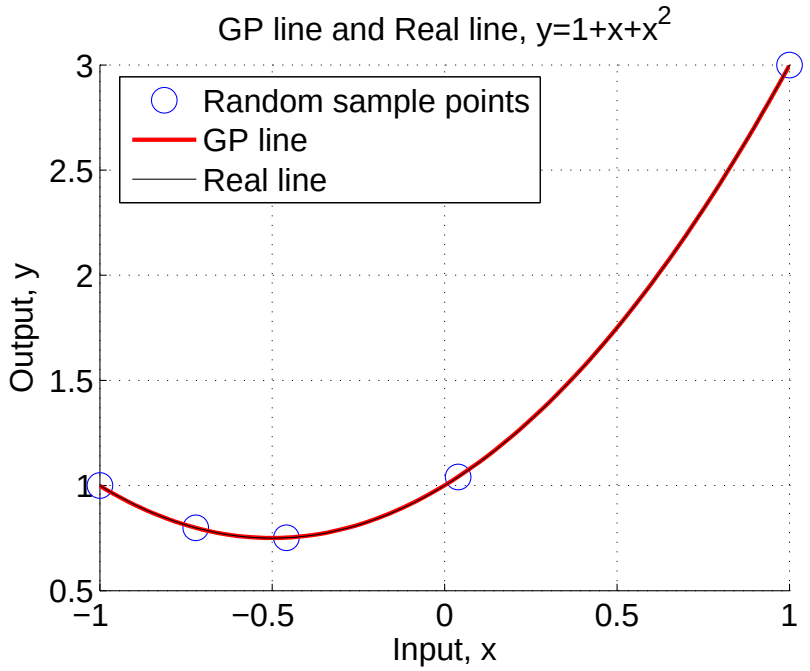

Figure 3.5: GP result and real function comparison on 1-D simple Legendre function.

It is noticeable here that the number of samples used in different methods are different. Only 5 samples are used in both compressed sensing approach with Legendre basis and Fourier basis. However, 50 samples are used in the GP approach. If the number of sample used in GP approach is 5, it could not even give a reasonable function compared the real function. Even if we put 50 samples into the GP method, it still could not give an accurate function compared to the real function. A small part of GP function result is being enlarged to show the difference between the result and the real function, which is shown in figure 3.15. From the figures, we can see the compressed sensing method with Fourier series basis is better than the other two methods in this simple Fourier function example. It is obvious to see that, if we could have the same basis for compressed sensing approach as the basis used in the function, our method works well. However, if we apply the wrong basis to the function, our method does not work.

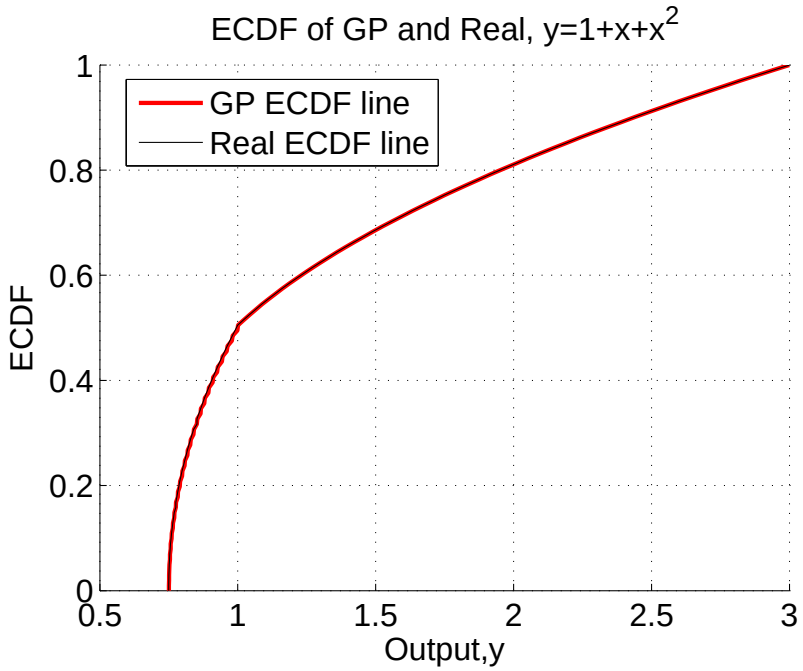

Figure 3.6: ECDF of GP approach result and real function on 1-D simple Legendre function.

3.4 2-D Legendre polynomials function

The function we seek to develop a surrogate approximation for in this demonstration is:

$$f(x_1, x_2) = 1 + x_1 + \frac{1}{2}(3x_2^2 - 1) + x_1 x_2, \tag{3.5}$$

where $(x_1, x_2) \in [-1, 1]^2$. This function has four modes in the Legendre basis spanned by the tensor product of the one dimensional Legendre bases for each individual dimension and no other components. Therefore, this function is known to be 4-sparse in the Legendre basis, and should be approximated exactly with very few function evaluations by our approach. We estimate the subfunctions, $\hat{f}_1$ and $\hat{f}_2$, as shown in figure 3.17. Here, the exact signal for each, (e.g., $f_1 + \hat{f}_0$ for the x_1 component) is also plotted on each figure.

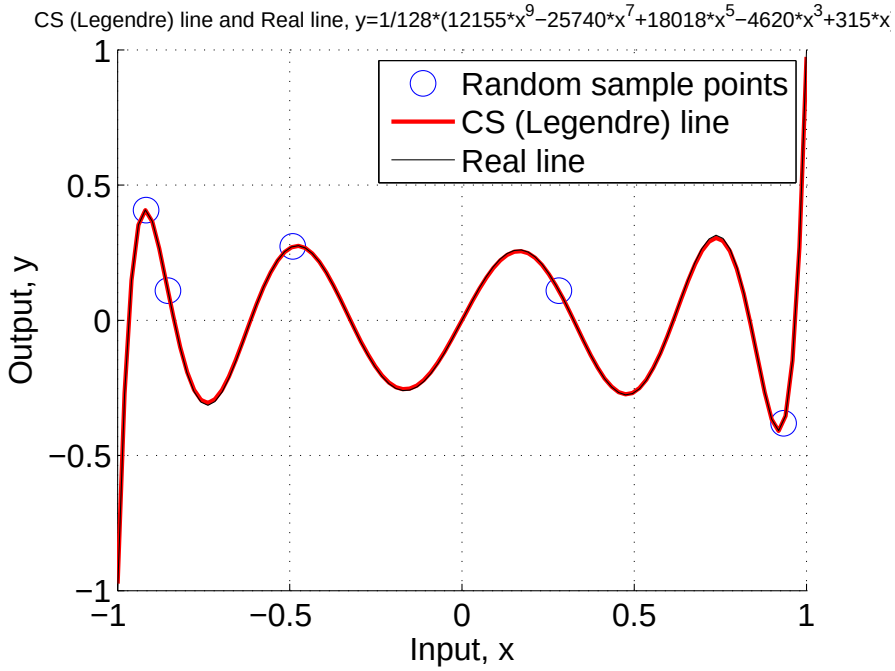

Figure 3.7: Compressed sensing approach with Legendre basis result and real function comparison on 1-D complicated Legendre function.

The approximations of these subfunctions are exact, thus, only one curve is seen in each plot on the figure. In this case, $\hat{f}_0$ is computed with $(x_1, x_2) = (0, 0)$. The construction of each subfunction required 5 evaluations of the function, $f(x_1, x_2)$ each. To determine this, we began with 2 samples for each subfunction construction independently and computed $\left\|\hat{c}_j\right\|_0$ for each input as described in chapter 2. We continued with this process for 3, 4, and 5 samples, where it was revealed that the $\left\|\hat{c}_j\right\|_0$ terms for each input were no longer varying increasing sample size. Thus, the process was stopped at 5 samples each. The third plot in the figure shows the validation error, $\|e\|_2$, which was computed for each successive approximation with 10 randomly chosen values of (x_1, x_2) that were propagated through each approximation, as well as the full function, $f(x1, x2)$. The first error value is the error from using just $\hat{f}_0$ to estimate $f(x_1, x_2)$. The second error value is associated with using $\hat{f}_1$, and the last error value is associated with using $\hat{f}_1 + \hat{f}_2 - \hat{f}_0$. As can be

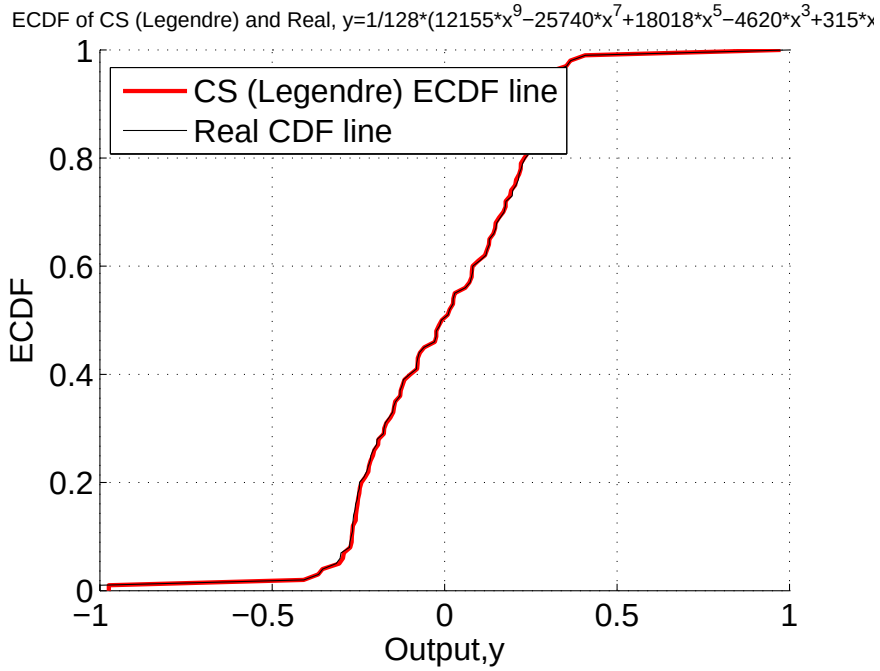

Figure 3.8: ECDF of compressed sensing approach with Legendre basis function and real function on 1-D complicated Legendre function.

seen from the plot, the error is decreasing as we add more terms to the approximation, but we still have not recovered the full signal from just the one input subfunctions. This can also be seen in the fourth plot of figure 3.17, where the functions $f_{1,2} = f(x_1, x_2)$ and $\hat{f}_{1,2}$ are plotted as a function of (x_1, x_2). The surfaces differ because the $\hat{f}_{1,2}$ function cannot account for the interaction effect of x_1 and x_2 in $f_{1,2}$ with only additive terms of one input subfunctions. We note that thus far, we have evaluated $f(x_1, x_2)$ a total of 20 times. These evaluations consist of 5 for each one input subfunction construction, and 10 for validation.

The next step in our modeling process then, is to add the two input subfunction of (x_1, x_2) following the procedure described in chapter 2. The result of adding the two input subfunction to our original surrogate representation is shown in figure 3.18. The top plot overlays the surrogate approximation, $\hat{f}_{x1,x2}$, and the full model, $f(x_1, x_2)$. The surrogate model is an exact match in this case. The bottom plot shows the ECDF of $f(x1, x2)$,

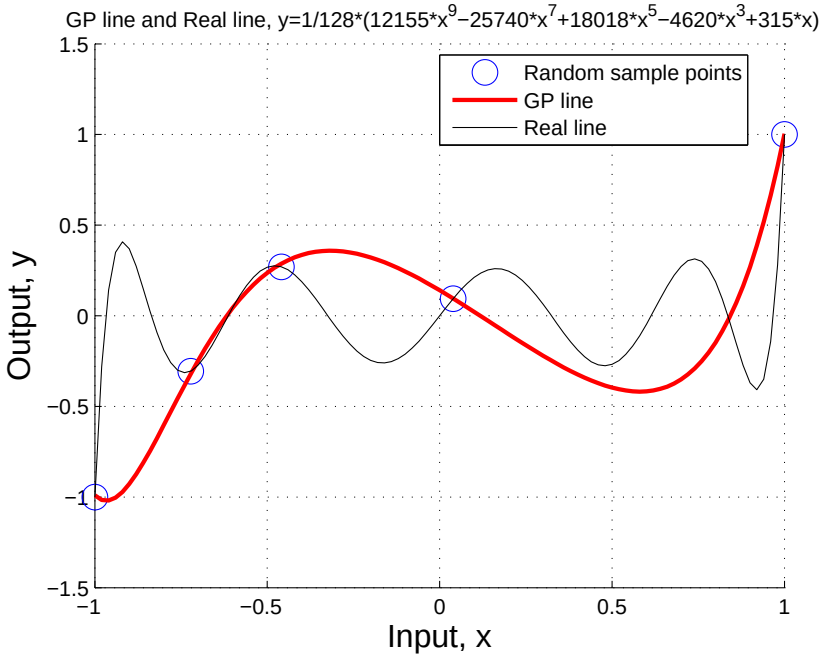

Figure 3.9: GP result and real function comparison on 1-D complicated Legendre function.

with $X_1 \sim U[0,1]$ and $X_2 \sim U[-0.5,0.5]$. This ECDF was estimated via Monte Carlo simulation using 1000 full model evaluations and is shown as a black line. Also shown on this plot is the ECDF of $\hat{f}_{x_1,x_2}$, which was estimated using 1000 samples of the surrogate approximation. This ECDF is shown as a dashed red line and matches the true ECDF exactly. To construct the surrogate approximation required only 25 full model evaluations, plus another 10 full model evaluations for validation. We use the tensor product of the previously computed sample points when constructing the two input subfunction, which allows us to reuse the 10 full model evaluations acquired in the construction of the one input subfunctions.

After implementation of compressed sensing approach with Legendre basis, GP approach is being applied to the same function. As 25 random sample points are randomly selected from compressed sensing approach, we want to use the sample number of sam-

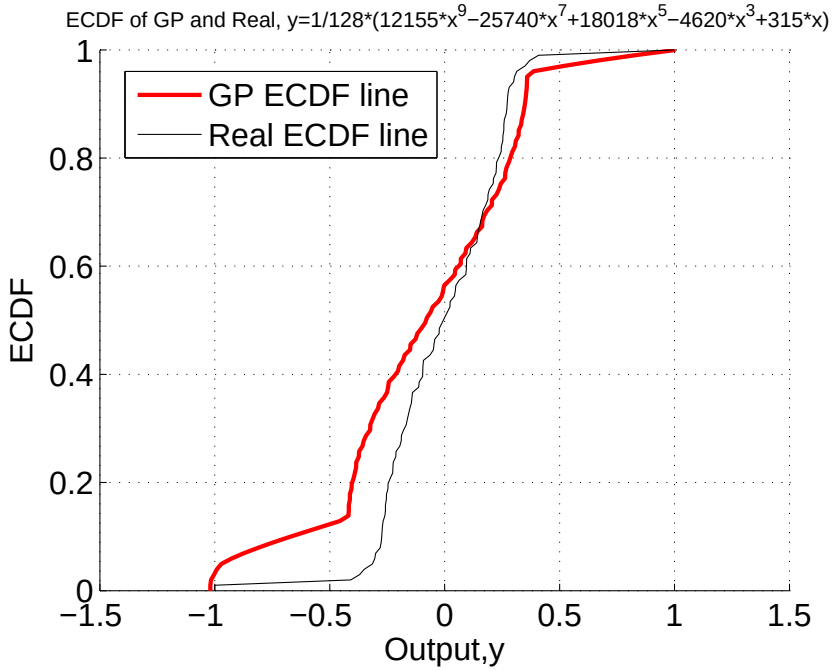

Figure 3.10: ECDF of Gaussian process result and real function on 1-D complicated Legendre function.

ple points in the GP approach for comparison. During experiment, we use 21 randomly selected sample points, with four sample points on the four vertexes of the sample space, which are: $(-1,-1)$, $(-1,1)$, $(1,-1)$ and $(1,1)$.

The result from compressed sensing approach and the real function are shown in figure 3.19. The ECDF of compressed sensing approach with Legendre basis result and real function are shown in figure 3.20. The result from GP approach and the real function are shown in figure 3.21. The ECDF of GP approach result and real function are shown in figure 3.22.

Although the result from compressed sensing approach with Legendre basis, result from GP approach and the real function all look similar in the graphs, actually compressed sensing approach with Legendre basis works better than GP approach in this example. 10,000 points are random generated from the sample space, the difference between the

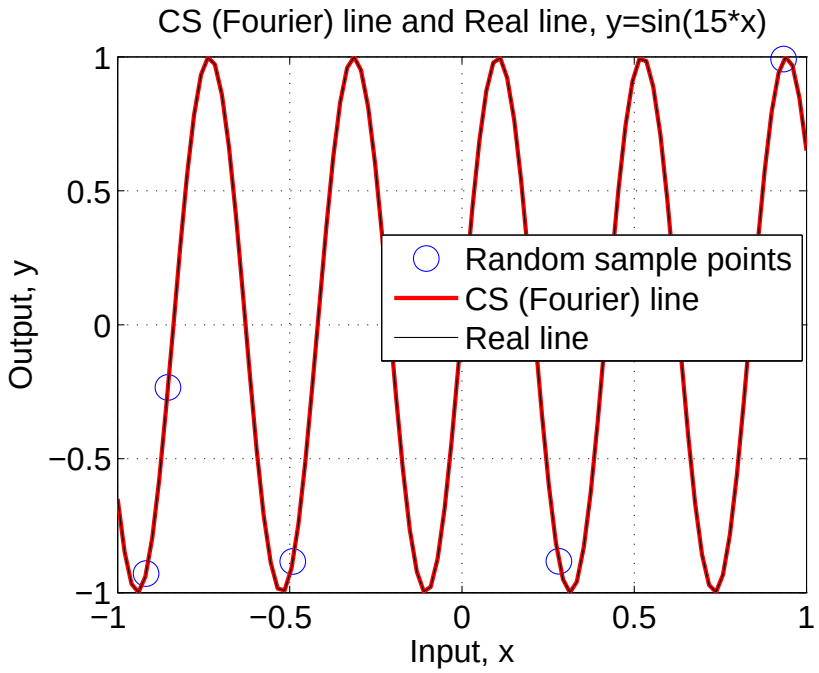

Figure 3.11: Compressed sensing approach with Fourier basis function and real function comparison on 1-D simple Fourier function.

two methods function result and the real function is calculated. The compressed sensing approach with Legendre basis gives a total norm two error as 5.2897e-04, but the GP appraoch gives a total norm two error as 0.7039, which is much more than the error from our approach.

3.5 Cooled gas turbine blade model

For this demonstration we consider a finite element model of a cooled gas turbine blade modeled after that provided in reference [72]. The blade profile and the random inputs to the finite element model are shown in figure 3.23. The computational model is a heat transfer model that simulates a cooled gas turbine blade in a hot gas path flow. The uncertain inputs to this system, their units, and their probability distributions are provided in table 3.1. Here, k is the thermal conductivity of the blade, T_{gas} is the external gas

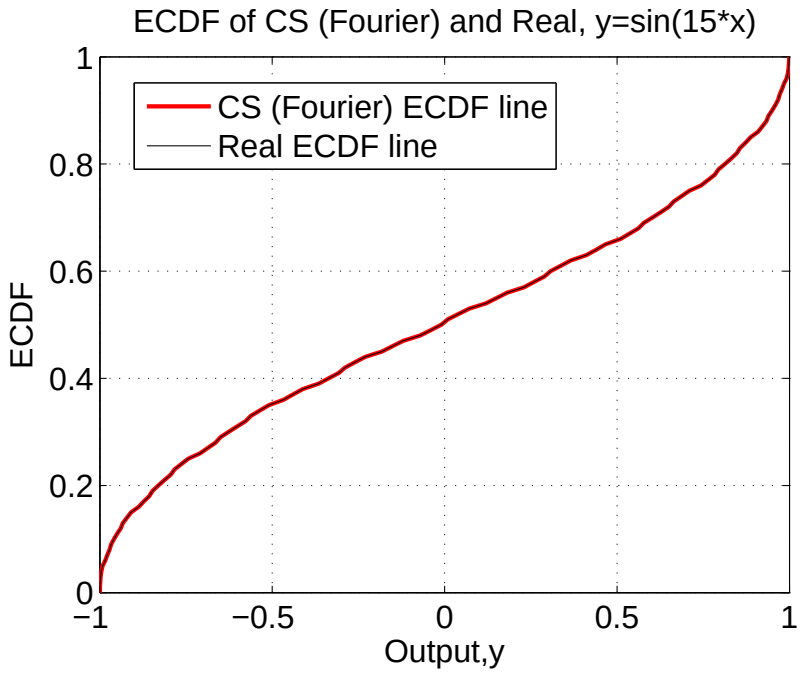

Figure 3.12: ECDF of compressed sensing approach with Fourier basis function and real function on 1-D simple Fourier function.

temperature, h_{LE} is the leading edge heat transfer coefficient, h_{TE} is the trailing edge heat transfer coefficient, T_{cool} is the cooling passage temperature, and h_{cool} is the cooling passage heat transfer coefficient. It is assumed that a failure will occur when the maximum temperature or temperature gradients become too large. This is modeled by combining damage due to high temperature and damage due to high temperature gradients as:

$$D := \frac{T_{max} - T_{max}^{des}}{T_{limit} - T_{max}^{des}} + \frac{|\nabla T|_{max} - |\nabla T|_{max}^{des}}{|\nabla T|_{limit} - |\nabla T|_{max}^{des}}, \tag{3.6}$$

where $T_{limit} = 1,500K$ is the limiting value of the temperature, $|\nabla T|_{limit} = 80,000K/m$ is the limiting value of the temperature $T_{max}^{des} = 1,430K$ is the maximum temperature at design intent conditions, and $|\nabla T|_{max}^{des} = 70,000K/m$ is the maximum temperature gradient at design-intent conditions. We consider the damage, D, as the quantity of interest

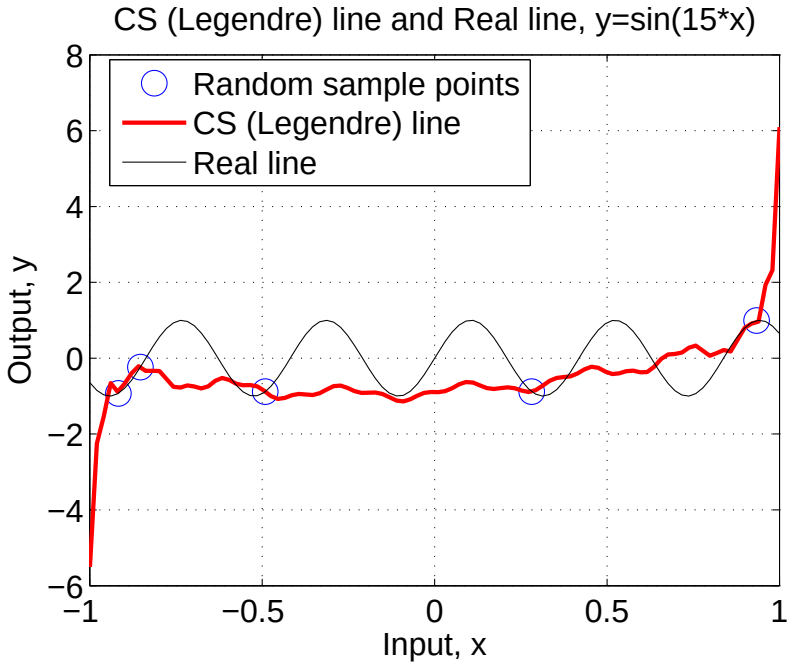

Figure 3.13: Compressed sensing approach with Legendre basis result and real function comparison on 1-D simple Fourier function.

Table 3.1: Cooled gas turbine blade model inputs and distribution, where $T(a, b, c)$ represents a triangular distribution with lower limit, a, mode, b, and upper limit, c.

Input	Units	Distribution
k	$W/(mK)$	T(28.5,30,31.5)
T_{gas}	K	T(1400,1500,1600)
h_{LE}	$W/(m^2K)$	T(15000,16000,17000)
h_{TE}	$W/(m^2K)$	T(3500,4000,4500)
T_{cool}	K	T(550,600,650)
h_{cool}	$W/(m^2K)$	T(1400,1500,1600)

of this system. Thus, our goal is to efficiently propagate uncertainty from the inputs to this particular output.

Using the methodology outlined in the 2nd chapter, we construct a compressed sensing surrogate representation of the damage output of the computational model of the cooled

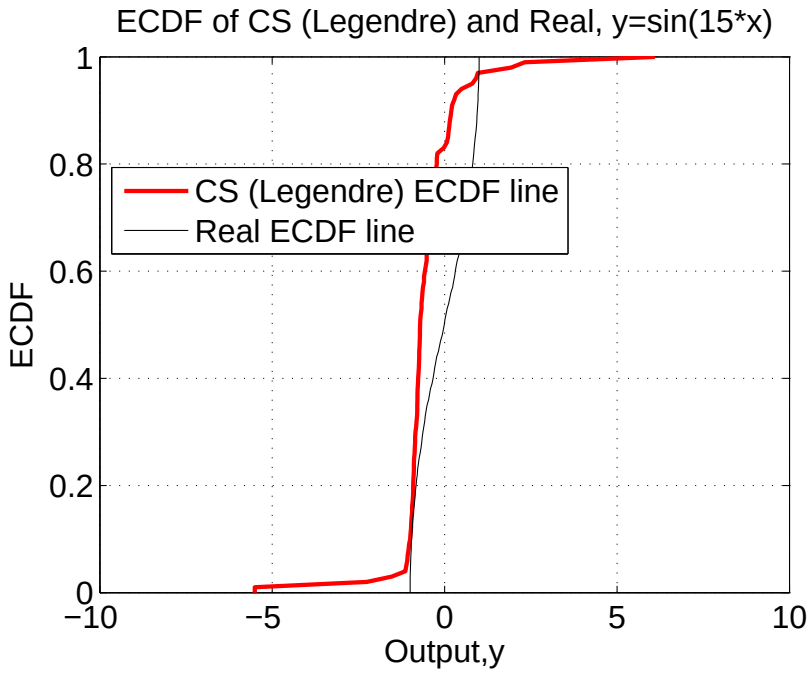

Figure 3.14: ECDF of compressed sensing approach with Legendre basis result and real function on 1-D simple Fourier function.

gas turbine blade. We construct this surrogate as an approximately additive function of the inputs using a sparse representation in the Legendre polynomial basis. After initially creating a purely additive surrogate (i.e., containing only the one input subfunctions), the approximate sensitivity indices, $\hat{S}_j$, for $j = 1, 2, ..., 6$, were computed according to equation 2.12. The results of this computation are presented in table 3.2. According to our greedy strategy for incorporating higher order subfunctions, which was described in the 2nd chapter, we would first consider adding the two input subfunction of (T_{gas}, T_{cool}), followed by the two input subfunction of (T_{gas}, h_{cool}), and so on.

In figure 3.24, the results for uncertainty propagation through the surrogate representation containing all subfunctions of one variable and five subfunctions of two inputs, along with the results from Monte Carlo simulation with the full computational model are shown. The two input subfunctions included in the surrogate representation are the pairs of T_{gas}

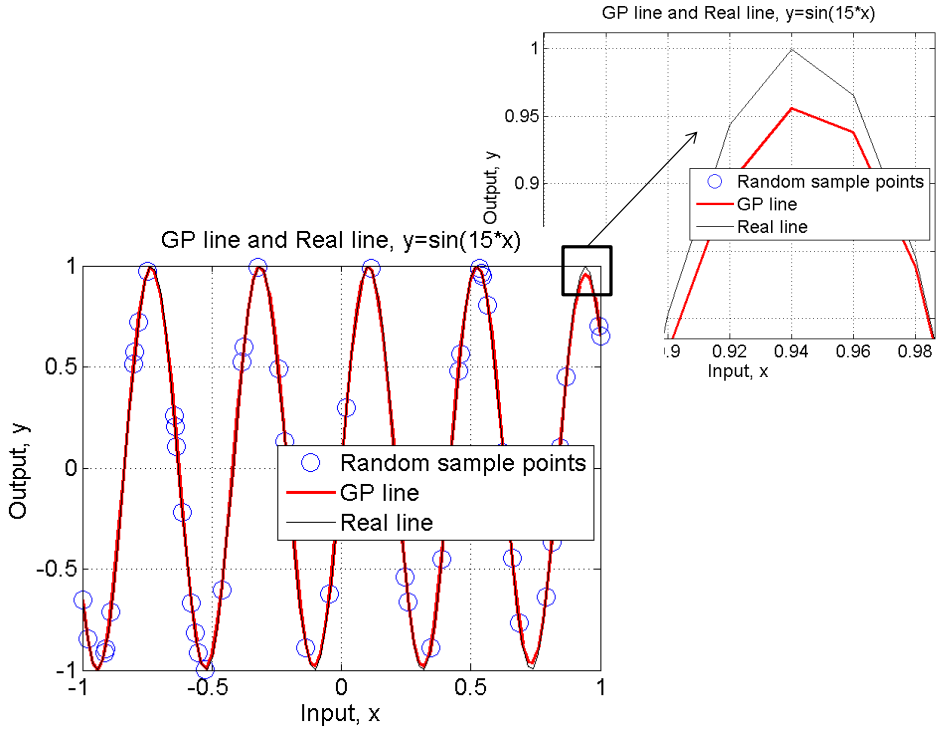

Figure 3.15: GP result and real function comparison on 1-D simple Fourier function.

Table 3.2: Approximate main effect sensitivity indices computed using the purely additive surrogate model.

Input	k	T_{gas}	h_{LE}	h_{TE}	T_{cool}	h_{cool}
$\hat{S}$	0.01	0.93	0.01	0.01	0.03	0.02

and each other variable. In figure 3.24, the top plot displays the histogram of Damage as computed by a 1,000 sample Monte Carlo simulation of the full model. The middle plot presents the histogram computed by sampling from the surrogate representation 1,000 times. The surrogate representation itself required only 115 full model evaluations, which consisted of 5 evaluations for each one input subfunction, 15 additional function

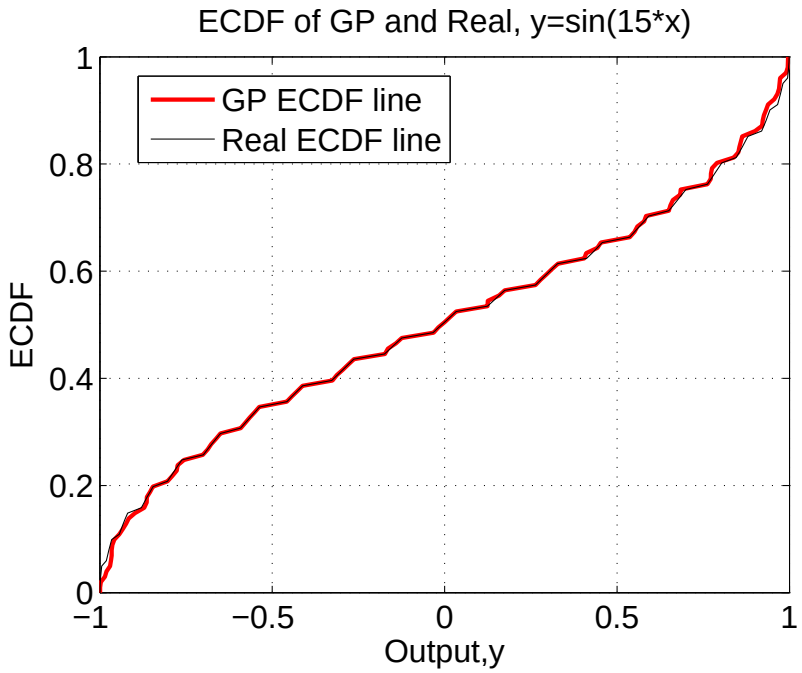

Figure 3.16: ECDF of GP approach result and real function on 1-D simple Fourier function.

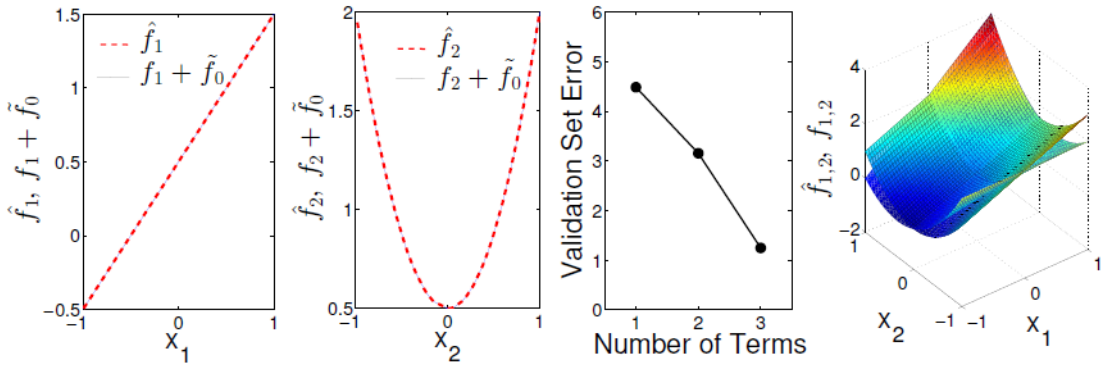

Figure 3.17: Subfunction of input X_1 and its sparse approximation (left plot) subfunction of input X_2 and its sparse approximation (second left plot), validation error as a function of the number of terms in the surrogate approximation (second from the right plot), and the full function and its additive surrogate approximation.

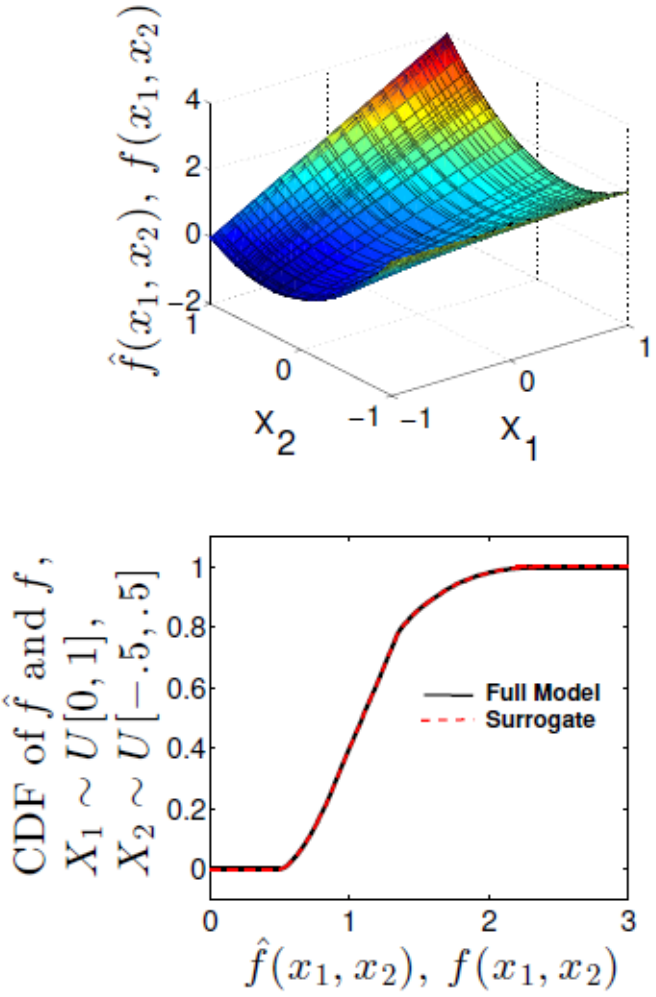

Figure 3.18: Surrogate approximation $\hat{f}$ and full model f (Top), empirical CDF function of full model via Monte Carlo simulation (black) and surrogate approximation (red) (bottom).

evaluations for each two input subfunction included, and 10 full model evaluations for the validation set. As can be seen from the two histograms, the surrogate model provides an accurate representation of the full model. In the bottom plot of the figure, the empirical cumulative distribution function for the Monte Carlo simulation of the full model and the surrogate representation are plotted. As is clear from the plot, these functions are nearly identical.

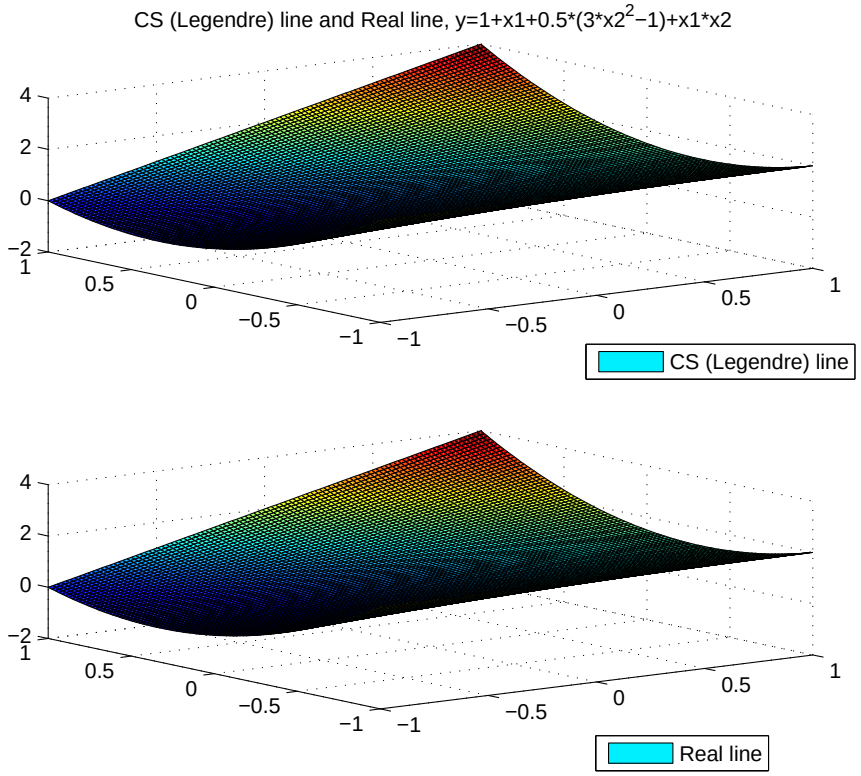

Figure 3.19: Compressed sensing approach with Legendre basis result and real function on 2-D Legendre polynomials function.

While the results from the surrogate representation constructed from 115 points and 12 terms (the offset term, 6 one input subfunctions, and 5 two input subfunctions) are excellent and represent substantial computational savings, more could have been saved depending on the desired threshold for the validation error. As we did in the case of the analytical function in the previous example, we randomly sampled 10 points in the input space, propagated them through the full model, and used the results as a validation set to test against with our surrogate representations. The results of this test as we added more terms to our approximation are shown in figure 3.25. The figure clearly shows that the validation error did not decrease anymore after the first 7 terms (the offset and the one

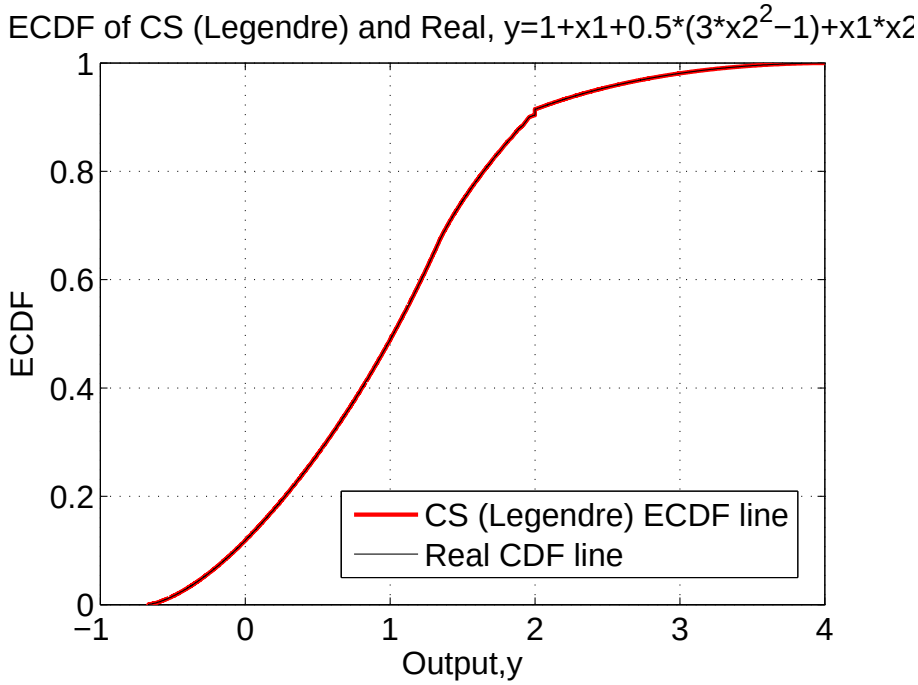

Figure 3.20: ECDF of compressed sensing approach with Legendre basis result and real function on 2-D Legendre polynomials function.

input subfunctions). Indeed, the results shown in figure 3.24 can be reproduced just as well with only the first 7 terms of the surrogate representation, which required only 40 full model evaluations to construct (including the 10 for the validation set). Had a reasonable threshold for the validation error been set, we would not have greedily added the two input subfunctions and the surrogate representation would have been essentially identical. The development of robust algorithms for constructing sparse surrogate representations that incorporate such heuristics is a topic for future work.

3.6 Uncertainty information example

In this section, a 1-D simple Legendre function is used to demonstrate of some possible methods to build uncertainty information for our compressed sensing method. The example used in this section is:

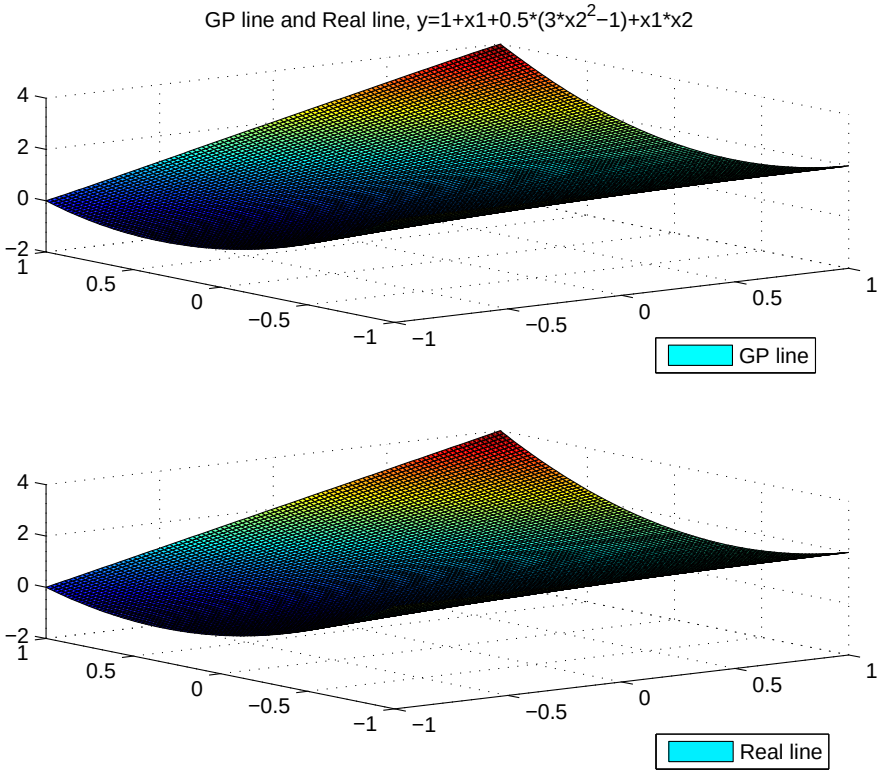

Figure 3.21: GP result on 2-D Legendre polynomials function.

$$f(x) = \frac{1}{2}(3x^2 - 1), \tag{3.7}$$

where $x \in [-5, 5]$, and this function is the third term in Legendre polynomials. As it is demonstrated in previous chapters and sections, using compressed sensing method with Legendre basis could solve this problem only within several model evaluations with very few randomly selected samples. However, in this section, the sample points are fixed at certain point for purpose of comparison and demonstration. The sample points are chosen as: $(-4.5, 29.8750)$, $(0, -0.5)$, $(1, 1)$, $(1.5, 2.8750)$, and $(4, 23.5)$. As a result, 4 different Brownian Bridge section are built from all adjacent points. Each Brownian Bridge building section would have 1000 randomly generated Brownian Bridges. It is shown in figure 3.26

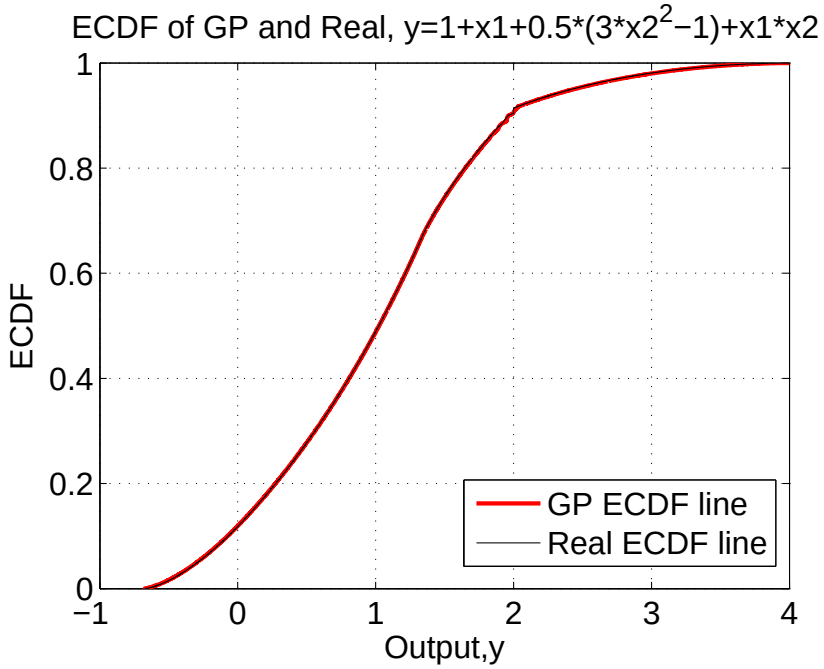

Figure 3.22: ECDF of GP approach result and real function on 2-D Legendre polynomials function.

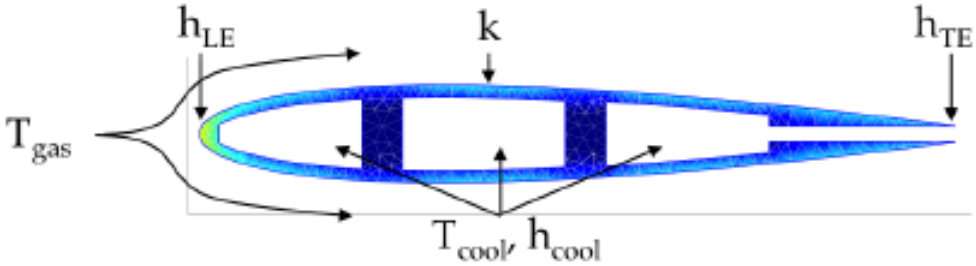

Figure 3.23: The cooled gas turbine blade profile and random input variables

and figure 3.27. All the Browian Bridge are built from uniform random walk Brownian motion.

The black lines in figure 3.26 and figure 3.27 are the randomly generated Brownian Bridges. The step size used in generating Brownian Bridges is 0.01. After using the

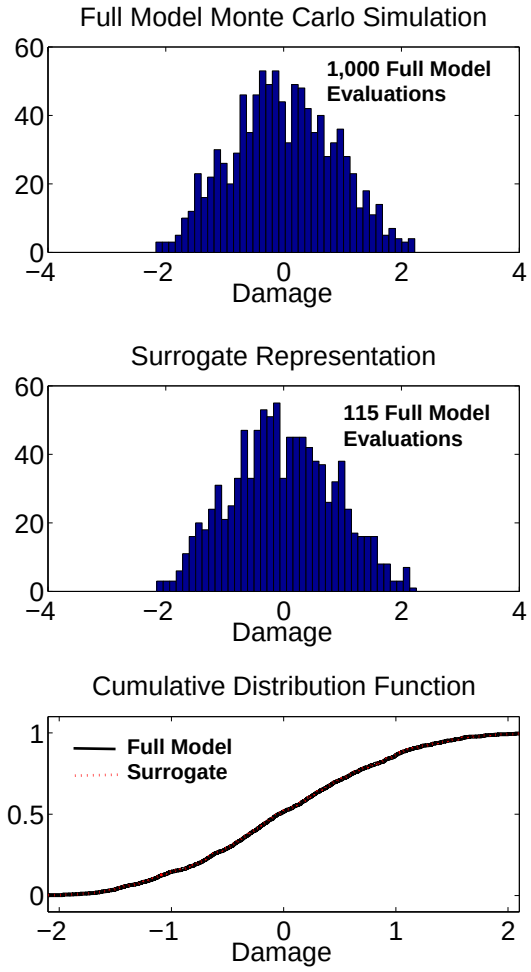

Figure 3.24: Histogram of full model Monte Carlo simulation (top), histogram of surrogate representation based Monte Carlo simulation (middle), empirical cumulative distributed functions of the full model and surrogate representation (bottom).

modified Lipschitz filtering approach and the modified VC-dimension filtering approach to filter original Brownian Bridges that do not satisfy the requirement mentioned in equation 2.32 and equation 2.41 separately, the red lines and the green lines are the ones which are still eligible separately. The angle used in modified Lipschitz filtering approach is 45 degrees, and the B used in modified VC-dimension filtering approach is 1.

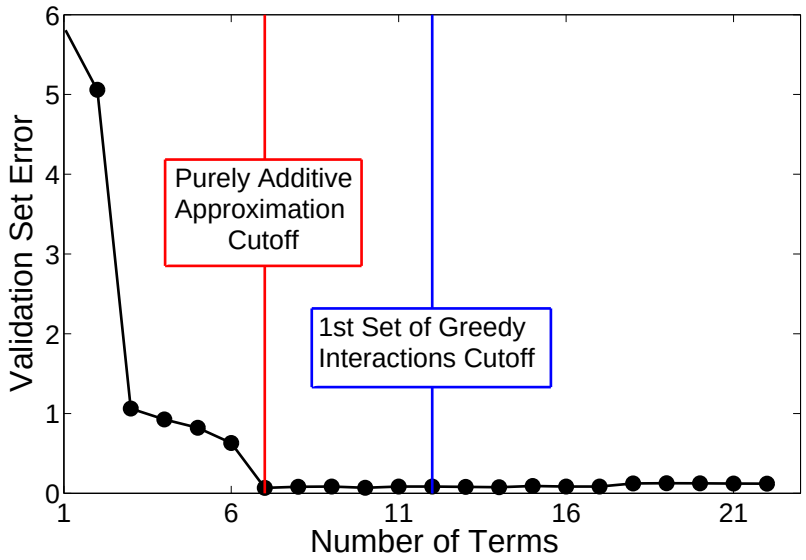

Figure 3.25: Validation error as a function of the number of terms in the surrogate approximation of the cooled gas turbine blade model. The red line indicates the end of the purely additive surrogate terms and the blue line indicates the end of the terms used in computed the results showing in figure 3.24.

The histogram for the modified Lipschitz filtering approach with $x = -3$ and $x = 3$ are shown in figure 3.28 and figure 3.29. And the histogram for the modified VC-dimension filtering approach with $x = -3$ and $x = 3$ are shown in figure 3.30 and figure 3.31.

There might be some concerns about those two graphs: figure 3.26 and figure 3.27. For some concave and convex functions, Brownian Bridge could have some possibility that they never have any intersections with the real function. However, changing step size of generating Brownian Bridges process would resolve this problem. With less step size, the Brownian Bridge would become more variable, the difference between the upper limit of all the Brownian Bridges and the lower limit would be larger. This would make sure that the Brownian Bridges would finally intersect with the real function while decreasing the step size.

For the figure using modified Lipschitz filtering approach, it is common to see with

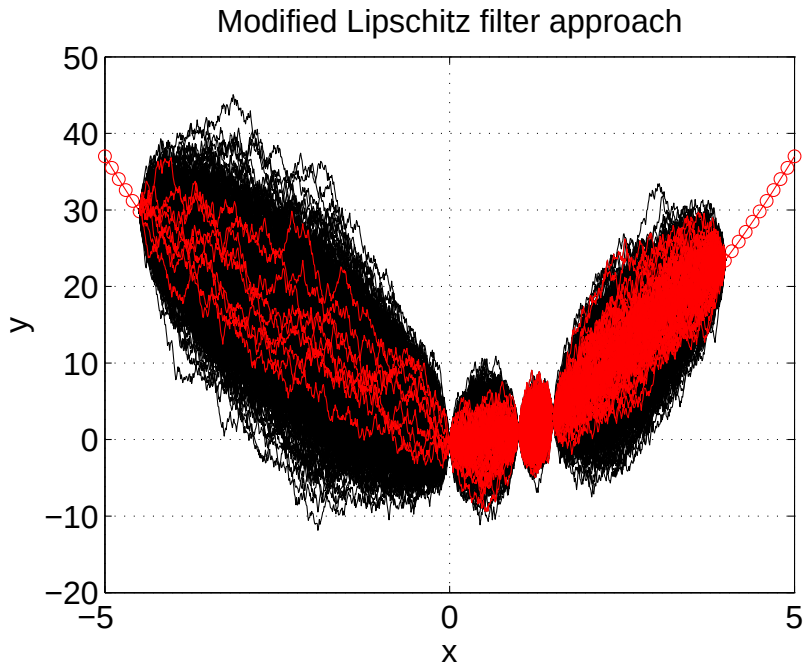

Figure 3.26: Brownian Bridges with modified Lipschitz approach. Black lines are for Browian Bridges, red lines are for eligible Brownian Bridges.

a large interval between Brownian Bridge starting point and ending point, this approach could filter many Brownian Bridges. This could be explained by equation 2.33. It is obvious to see the Brownian Bridge slope is related to the term $\frac{x_T}{T}$, where this term is related to the interval distance of the Brownian Bridge. The sum of uniform distribution would become IrwinHall distribution [73], with central limit theorem, it would become normal distribution [74]. In other words, when T or the interval distance become larger, most of the Brownian Bridges would have a smaller $\frac{x_T}{T}$ term, which increases the k_{BB}, and this could make modified Lipschitz filtering approach to filter more Brownian Bridges.

For the figure using modified VC-dimension filtering approach, it is common to see with a small interval between Brownian Bridge starting point and ending point, this approach could hardly filter any Brownian Bridges. This could be explained by the method mentioned in ending part in section 2.8. When the compared Brownian Bridge have a

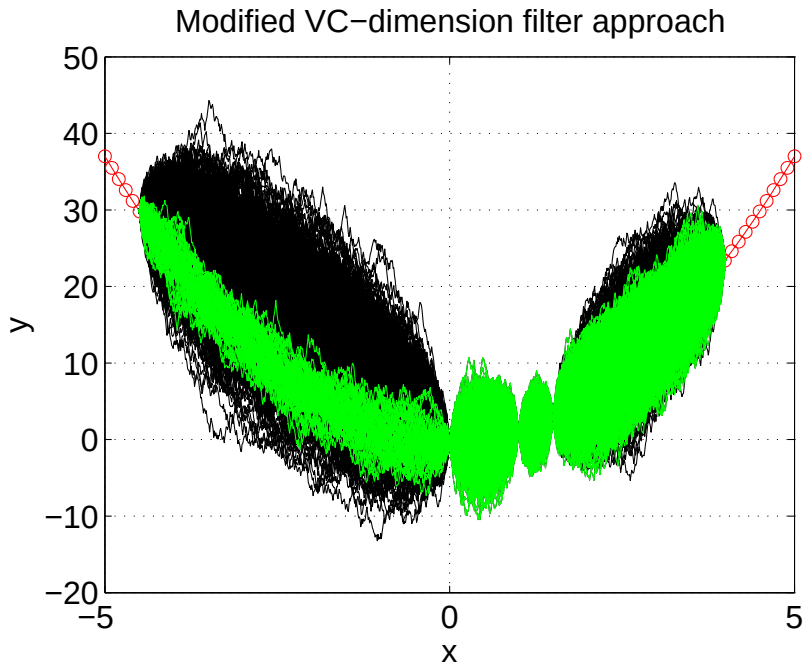

Figure 3.27: Brownian Bridges with modified VC-dimension approach. Black lines are for Browian Bridges, green lines are for eligible Brownian Bridges.

small input range, this would generate a relatively small estimation for value d_2. As d_2 could always be less than d_1 on small intervals, it is possible that modified VC-dimension filtering approach never filter any Brownian Bridges.

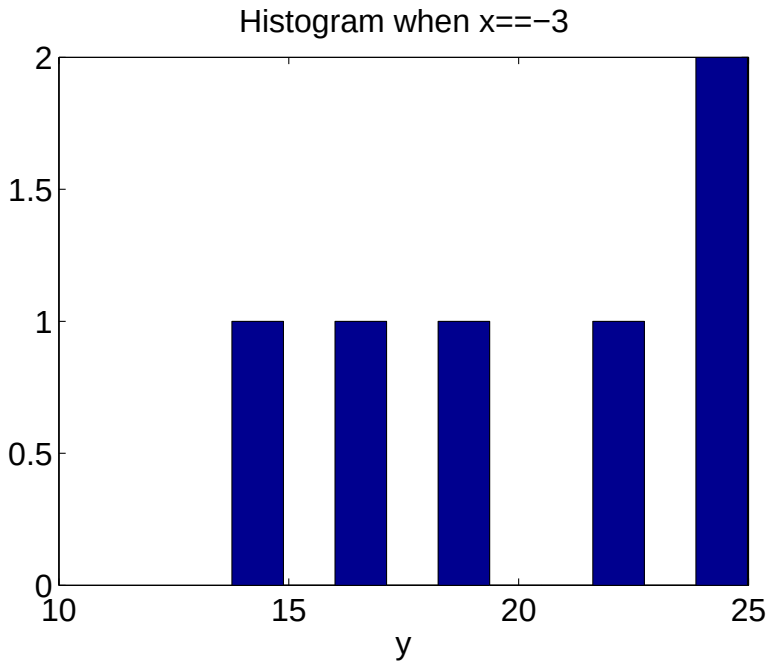

Figure 3.28: Histogram of eligible Brownian Bridges when $x = -3$ with modified Lipschitz approach.

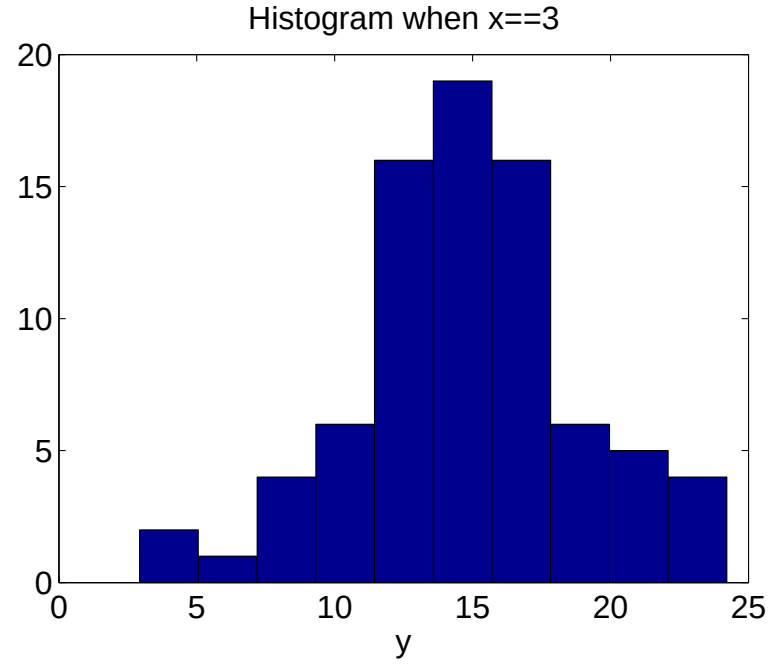

Figure 3.29: Histogram of eligible Brownian Bridges when $x = 3$ with modified Lipschitz approach.

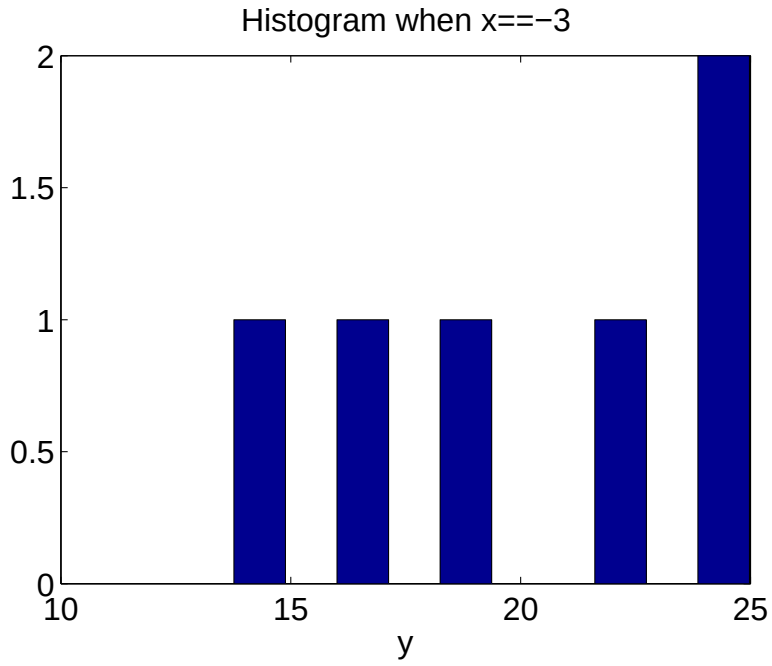

Figure 3.30: Histogram of eligible Brownian Bridges when $x = -3$ with modified VC-dimension approach.

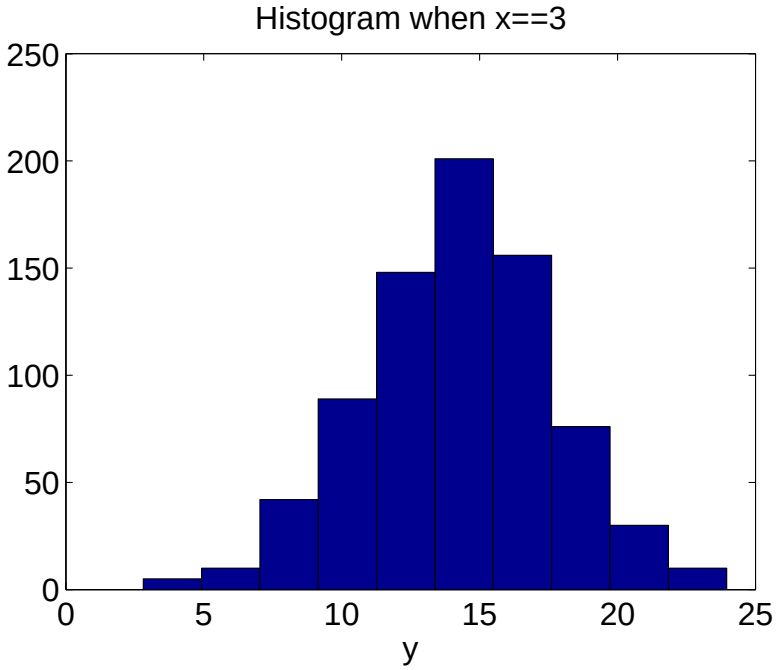

Figure 3.31: Histogram of eligible Brownian Bridges when $x = 3$ with modified VC-dimension approach.

4. SUMMARY AND CONCLUSIONS *

We have presented a compressed sensing approach to uncertainty propagation by constructing a sparse surrogate representation of a computational model. The approach relied on the assumption of approximate additivity. We demonstrated our approach on some approximately additive functions that was known to be sparse in the Legendre basis and Fourier basis, and the results revealed the effectiveness of our approach under perfect circumstances. We made a comparison with GP regression method using the same analytical functions. We also demonstrated our approach on a finite element model of a cooled gas turbine blade for which we had no pre-requisite knowledge of sparsity in the Legendre basis or approximate additivity. We propose to build the corresponding uncertainty information for our method with Brownian Bridges, and some possible approaches are used to filter unreasonable function lines. Nevertheless, our approach yielded significant computational savings as compared to Monte Carlo simulation with negligible loss of accuracy in the uncertainty propagation results. We hypothesize that many other engineering computational models are approximately additive and can be represented well in a polynomial basis (or other bases as necessary). If this hypothesis is true, then the concept of using compressed sensing to efficiently find a sparse representation of the underlying function is promising. To ensure the exploitation of sparsity and approximate additivity in these situations, robust algorithms that identify a good basis to search for sparseness in, and when a good sparse representation has been found in that basis, are necessary. We have provided several heuristics in this work that have some potential for providing this robustness, and their study is a topic of future work.

*Reprinted with permission from "A compressed sensing approach to uncertainty propagation for approximately additive functions" by Kaiyu Li, Douglas Allaire, 2016. ASME 2016 International Design Engineering Technical Conferences, IDETC/CIE, Copyright 2016 by ASME.

REFERENCES

[1] K. Li and D. Allaire, "A compressed sensing approach to uncertainty propagation for approximately additive functions," *ASME 2016 International Design Engineering Technical Conferences, IDETC/CIE*, 2016.

[2] S. Brooks, "Markov chain monte carlo method and its application," *Journal of the royal statistical society: series D (the Statistician)*, vol. 47, no. 1, pp. 69–100, 1998.

[3] E. Zio, *The Monte Carlo simulation method for system reliability and risk analysis*. Springer, 2013.

[4] F.-C. Wu and Y.-P. Tsang, "Second-order monte carlo uncertainty/variability analysis using correlated model parameters: application to salmonid embryo survival risk assessment," *Ecological Modelling*, vol. 177, no. 34, pp. 393 – 414, 2004.

[5] A. Van der Waart, "Asymptotic statistics, volume 27 of cambridge series in statistical and probabilistic mathematics 03," 1998.

[6] J. Z. Ma and E. Ackerman, "Parameter sensitivity of a model of viral epidemics simulated with monte carlo techniques. ii. durations and peaks," *International journal of bio-medical computing*, vol. 32, no. 3, pp. 255–268, 1993.

[7] R. MacDonald, J. Campbell, *et al.*, "Valuation of supplemental and enhanced oil recovery projects with risk analysis," *Journal of petroleum technology*, vol. 38, no. 01, pp. 57–69, 1986.

[8] R. J. Eggert, "Design variation simulation of thick-walled cylinders," *Journal of Mechanical Design*, vol. 117, no. 2A, pp. 221–228, 1995.

[9] J. Rastegar and B. Fardanesh, "Geometric synthesis of manipulators using the monte carlo method," *Journal of Mechanical Design*, vol. 112, no. 3, pp. 450–452, 1990.

[10] S. Amaral, D. Allaire, and K. Willcox, "A decomposition-based approach to uncertainty analysis of feed-forward multicomponent systems," *International Journal for Numerical Methods in Engineering*, vol. 100, no. 13, pp. 982–1005, 2014.

[11] D. Allaire and K. Willcox, "Uncertainty assessment of complex models with application to aviation environmental policy-making," *Transport Policy*, vol. 34, pp. 109–113, 2014.

[12] D. Allaire, G. Noel, K. Willcox, and R. Cointin, "Uncertainty quantification of an aviation environmental toolsuite," *Reliability Engineering & System Safety*, vol. 126, pp. 14–24, 2014.

[13] G. Grimmett and D. Stirzaker, *Probability and random processes*. Oxford university press, 2001.

[14] J. C. Helton and F. J. Davis, "Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems," *Reliability Engineering & System Safety*, vol. 81, no. 1, pp. 23–69, 2003.

[15] H. Niederreiter, *Quasi-Monte Carlo Methods*. Wiley Online Library, 2010.

[16] R. E. Caflisch, "Monte carlo and quasi-monte carlo methods," *Acta numerica*, vol. 7, pp. 1–49, 1998.

[17] S. Au and J. L. Beck, "A new adaptive importance sampling scheme for reliability calculations," *Structural safety*, vol. 21, no. 2, pp. 135–158, 1999.

[18] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 42, no. 1, pp. 55–61, 2000.

[19] T. W. Simpson, D. K. Lin, and W. Chen, "Sampling strategies for computer experiments: design and analysis," *International Journal of Reliability and Applications*, vol. 2, no. 3, pp. 209–240, 2001.

[20] R. Jin, W. Chen, and A. Sudjianto, "On sequential sampling for global metamodeling in engineering design," in *ASME 2002 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pp. 539–548, American Society of Mechanical Engineers, 2002.

[21] M. C. Kennedy and A. O'Hagan, "Bayesian calibration of computer models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 3, pp. 425–464, 2001.

[22] F. Pérez-Cruz, S. Van Vaerenbergh, J. J. Murillo-Fuentes, M. Lázaro-Gredilla, and I. Santamaria, "Gaussian processes for nonlinear signal processing: An overview of recent advances," *IEEE Signal Processing Magazine*, vol. 30, no. 4, pp. 40–50, 2013.

[23] J. Ko and D. Fox, "Gp-bayesfilters: Bayesian filtering using gaussian process prediction and observation models," *Autonomous Robots*, vol. 27, no. 1, pp. 75–90, 2009.

[24] M. Jadaliha, Y. Xu, J. Choi, N. S. Johnson, and W. Li, "Gaussian process regression for sensor networks under localization uncertainty," *IEEE Transactions on Signal Processing*, vol. 61, no. 2, pp. 223–237, 2013.

[25] J. Quinonero-Candela, C. E. Rasmussen, and C. K. Williams, "Approximation methods for gaussian process regression," *Large-scale kernel machines*, pp. 203–224, 2007.

[26] M. S. Eldred, A. A. Giunta, S. S. Collis, N. Alexandrov, R. Lewis, *et al.*, "Second-order corrections for surrogate-based optimization with model hierarchies," in *Pro-*

ceedings of the 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, NY,, Aug, pp. 2013–2014, 2004.

[27] D. Allaire and K. Willcox, "Surrogate modeling for uncertainty assessment with application to aviation environmental system models," *AIAA journal*, vol. 48, no. 8, pp. 1791–1803, 2010.

[28] A. C. Antoulas, *Approximation of large-scale dynamical systems*, vol. 6. Siam, 2005.

[29] J. M. Pasini and T. Sahai, "Polynomial chaos based uncertainty quantification in hamiltonian and chaotic systems," in *52nd IEEE Conference on Decision and Control*, pp. 1113–1118, IEEE, 2013.

[30] H. N. Najm, "Uncertainty quantification and polynomial chaos techniques in computational fluid dynamics," *Annual Review of Fluid Mechanics*, vol. 41, pp. 35–52, 2009.

[31] M. Villegas, F. Augustin, A. Gilg, A. Hmaidi, and U. Wever, "Application of the polynomial chaos expansion to the simulation of chemical reactors with uncertainties," *Mathematics and Computers in Simulation*, vol. 82, no. 5, pp. 805–817, 2012.

[32] K. Tang, P. M. Congedo, and R. Abgrall, "Sensitivity analysis using anchored anova expansion and high-order moments computation," *International Journal for Numerical Methods in Engineering*, vol. 102, no. 9, pp. 1554–1584, 2015.

[33] K. Tang, P. M. Congedo, and R. Abgrall, "Adaptive surrogate modeling by anova and sparse polynomial dimensional decomposition for global sensitivity analysis in fluid simulation," *Journal of Computational Physics*, vol. 314, pp. 557–589, 2016.

[34] W. Yao, X. Chen, W. Luo, M. van Tooren, and J. Guo, "Review of uncertainty-based multidisciplinary design optimization methods for aerospace vehicles," *Progress in Aerospace Sciences*, vol. 47, no. 6, pp. 450–479, 2011.

[35] X. S. Gu, J. E. Renaud, and C. L. Penninger, "Implicit uncertainty propagation for robust collaborative optimization," *Journal of Mechanical Design*, vol. 128, no. 4, pp. 1001–1013, 2006.

[36] M. McDonald and S. Mahadevan, "Uncertainty quantification and propagation for multidisciplinary system analysis," in *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Victoria, BC, Canada, Sep*, pp. 9–12, 2008.

[37] M. Kokkolaras, Z. P. Mourelatos, and P. Y. Papalambros, "Design optimization of hierarchically decomposed multilevel systems under uncertainty," in *ASME 2004 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pp. 613–624, American Society of Mechanical Engineers, 2004.

[38] S. Mahadevan and N. Smith, "Efficient first-order reliability analysis of multidisciplinary systems," *International Journal of Reliability and Safety*, vol. 1, no. 1-2, pp. 137–154, 2006.

[39] S. J. Julier, "The scaled unscented transformation," in *Proceedings of the 2002 American Control Conference (IEEE Cat. No. CH37301)*, vol. 6, pp. 4555–4559, IEEE, 2002.

[40] H. Liu, W. Chen, M. Kokkolaras, P. Y. Papalambros, and H. M. Kim, "Probabilistic analytical target cascading: a moment matching formulation for multilevel optimization under uncertainty," *Journal of Mechanical Design*, vol. 128, no. 4, pp. 991–1000, 2006.

[41] M. M. Putko, A. C. Taylor, P. A. Newman, and L. L. Green, "Approach for input uncertainty propagation and robust design in cfd using sensitivity derivatives," *Journal of Fluids Engineering*, vol. 124, no. 1, pp. 60–69, 2002.

[42] X. Du and W. Chen, "A most probable point-based method for efficient uncertainty analysis," *Journal of Design and Manufacturing Automation*, vol. 4, no. 1, pp. 47–66, 2001.

[43] S. Rahman and D. Wei, "A univariate approximation at most probable point for higher-order reliability analysis," *International journal of solids and structures*, vol. 43, no. 9, pp. 2820–2839, 2006.

[44] I. M. Sobol, "Theorems and examples on high dimensional model representation," *Reliability Engineering & System Safety*, vol. 79, no. 2, pp. 187–193, 2003.

[45] H. Rabitz and Ö. F. Aliş, "General foundations of high-dimensional model representations," *Journal of Mathematical Chemistry*, vol. 25, no. 2-3, pp. 197–233, 1999.

[46] X. Ma and N. Zabaras, "An adaptive high-dimensional stochastic model representation technique for the solution of stochastic partial differential equations," *Journal of Computational Physics*, vol. 229, no. 10, pp. 3884–3915, 2010.

[47] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola, *Global sensitivity analysis: the primer*. John Wiley & Sons, 2008.

[48] M. Griebel and M. Holtz, "Dimension-wise integration of high-dimensional functions with applications to finance," *Journal of Complexity*, vol. 26, no. 5, pp. 455–489, 2010.

[49] X. Wang and K.-T. Fang, "The effective dimension and quasi-monte carlo integration," *Journal of Complexity*, vol. 19, no. 2, pp. 101–124, 2003.

[50] X. Wang and I. H. Sloan, "Why are high-dimensional finance problems often of low effective dimension?," *SIAM Journal on Scientific Computing*, vol. 27, no. 1, pp. 159–183, 2005.

[51] D. L. Donoho, "Compressed sensing," *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[52] E. J. Candes and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?," *IEEE transactions on information theory*, vol. 52, no. 12, pp. 5406–5425, 2006.

[53] M. F. Duarte and Y. C. Eldar, "Structured compressed sensing: From theory to applications," *IEEE Transactions on Signal Processing*, vol. 59, no. 9, pp. 4053–4085, 2011.

[54] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on information theory*, vol. 52, no. 2, pp. 489–509, 2006.

[55] L. Brutman, "Lebesgue functions for polynomial interpolation-a survey," *Annals of Numerical Mathematics*, vol. 4, pp. 111–128, 1996.

[56] H. Rauhut and R. Ward, "Sparse legendre expansions via 1-minimization," *Journal of approximation theory*, vol. 164, no. 5, pp. 517–533, 2012.

[57] I. V. Oseledets, "Tensor-train decomposition," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, 2011.

[58] G. B. Arfken, *Mathematical methods for physicists. 3rd ed. George Arfken.* Orlando : Academic Press, [1985], 1985.

[59] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, vol. 14, pp. 1137–1145, 1995.

[60] W. Chen, R. Jin, and A. Sudjianto, "Analytical variance-based global sensitivity analysis in simulation-based design under uncertainty," *Journal of mechanical design*, vol. 127, no. 5, pp. 875–886, 2005.

[61] A. J. Eggers Jr, H. J. Allen, and S. E. Neice, "A comparative analysis of the performance of long-range hypervelocity vehicles," 1955.

[62] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning, The MIT Press, 2006.

[63] M. Ebden, "Gaussian processes for regression: A quick introduction," *The Website of Robotics Research Group in Department on Engineering Science, University of Oxford*, 2008.

[64] R. M. Dudley, *Real analysis and probability*, vol. 74. Cambridge University Press, 2002.

[65] I. Karatzas and S. Shreve, *Brownian motion and stochastic calculus*, vol. 113. Springer Science & Business Media, 2012.

[66] F. B. Knight, "On the random walk and brownian motion," *Transactions of the American Mathematical Society*, vol. 103, no. 2, pp. 218–228, 1962.

[67] T. Antal and S. Redner, "Escape of a uniform random walk from an interval," *Journal of statistical physics*, vol. 123, no. 6, pp. 1129–1144, 2006.

[68] M. O'Searcoid, *Metric spaces*. Springer Science & Business Media, 2006.

[69] V. Koltun, "Advanced geometric algorithms lecture notes," 2006.

[70] V. Vapnik, E. Levin, and Y. Le Cun, "Measuring the vc-dimension of a learning machine," *Neural Computation*, vol. 6, no. 5, pp. 851–876, 1994.

[71] V. N. Vapnik and V. Vapnik, *Statistical learning theory*, vol. 1. Wiley New York, 1998.

[72] D. L. Darmofal, "16.901 computational methods in aerospace engineering, spring 2005," 2016.

[73] N. L. Johnson, S. Kotz, and N. Balakrishnan, *Continuous univariate distributions, vol. 2 of wiley series in probability and mathematical statistics: applied probability and statistics*. Wiley, New York,, 1995.

[74] E. R. Ziegel and J. Rice, *Mathematical Statistics and Data Analysis*. JSTOR, 1995.