Mathematical Foundations for Physics-Based Simulations

Larry A. Lambe *
Multidisciplinary Software Systems Research Corporation (MSSRC)
P.O. Box 6667†
Bloomingdale, IL 60108
llambe@mssrc.com

August 23, 2012

Contents

*The author is grateful to AFRL, WPAFB for SBIR Phase I & II support which enabled this research and to VKI, AFRL, and ONR for support during this Lecture Series

†Current address: PO Box 401, Blooimgdale, IL 60108

1 Introduction

A suitable subtitle to this lecture would be, "A Brief Excursion into Statistical Learning Theory". Hopefully, the sense of that will be clear at the end of the lecture.

There are many examples of scientific and engineering developments that have been and continue to be conducted using physics-based simulations. Such simulations were used at Los Alamos National Laboratory to study the behavior of nuclear weapons early on (Harlow and Metropolis, 1983), but the list goes on to include cosmological simulation (see e.g. Cosmological Simulations, http://hipacc.ucsc.edu/Bolshoi/index.html), weather prediction (see e.g. Climate Prediction, http://www.cpc.ncep.noaa.gov/) and many others including bioinformatics and computational fluid dynamics simulation.

Computer simulations can be used to study complex processes which would otherwise be more costly or would be unreasonably time consuming to study (or both). As such, computer simulations are based on a hopefully accurate approximation of the process one wishes to study. Such approximations are called mathematical or statistical models. The way one goes about deriving such models from a given process is the subject of this lecture.

Note that there are many different ways that model simulations may be derived for a given process. In this lecture, we will concentrate on a large class of methods that are called statistical modeling. Another class of methods that are noted in particular are given by what are called Monte Carlo methods. For those, see, for example, (Kalos and Whitlock, 2008; Rubinstein and Kroese, 2008).

1.1 From Data to Prediction: An Early Example

Early in the year 1801, the Italian astronomer Giuseppe Piazzi discovered Ceres and was able to track its path for 40 days before it was lost in the glare of the sun. Naturally, astronomers wanted to determine the position of Ceres after it emerged from behind the sun. Using the data Piazzi obtained, it would be possible to approximate that position by solving the nonlinear equations of Kepler for planetary motion – a complicated and, no doubt, error prone process at the time. Interestingly, Carl Friedrich Gauß had, in 1794, worked out a method that could be used to obtain an approximation to the orbit using the given data. That method is known today as ordinary linear least squares regression (OLLSR). Gauß did not publish his result until later (Gauss and Davis, 1857). In the meantime, others had independently formulated the least-squares method. See (Hald, 2007) for more information.

In the following, we will take a modern approach to deriving OLLSR and other statistical models. In fact, a unified framework will emerge that can be used to derive many familiar and possibly some unfamiliar models, but before that, we will briefly describe a second motivating example stemming from the work in (Kolonay and Lambe, 2012).

2 A Problem in Wing Design

An important problem in wing design is the minimization of induced drag. Such a problem is investigated in (Kolonay and Lambe, 2012) where the objective function is computed using an Euler solver. A problem of the form described in paragraph two of Section 1 arises. This is because numeric optimizers require objective function evaluations and the Euler solver can take many hours to compute just one function evaluation. The particular objective function in this case has twenty independent variables defined on the hypercube $[-10, 10]^{20} \times [-5, 10]$, a 21 dimensional space.

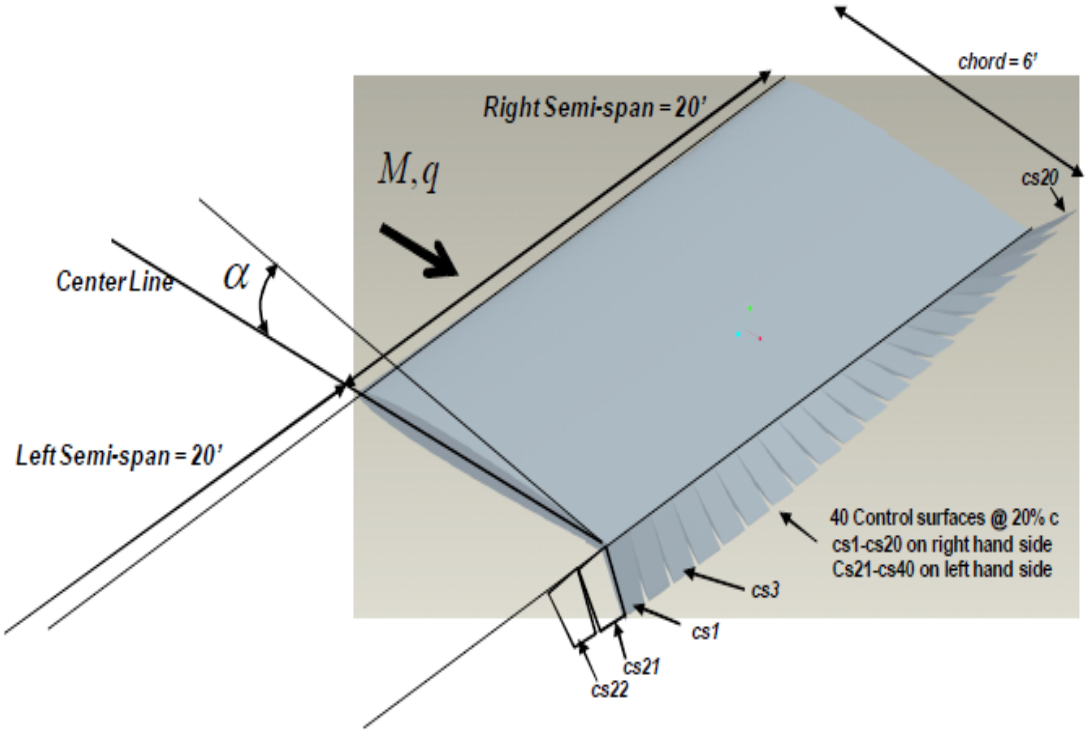

Figure 1: Goland Wing with 20 trailing edge control surfaces and the free stream angle of attack (α).

The problem to be solved is the following.

- Derive a statistical model from as few function evaluations as possible which can be used as a *surrogate* for the actual objective function.
- Obtain a "measure of goodness" that can be used to quantify the quality of the model outside the points used in deriving the model.

In the following Sections, this problem will be placed in a more general context. In all cases, the general problem of deriving the model can be stated as

- Starting with partial knowledge of a function,
- approximate the function (on some suitable domain) from that partial knowledge.

3 A Quick Review of Relevant Statistical Concepts

Any good probability and statistics textbook may be used as a reference for this Section.

A probability density is a function $\rho : \mathbb{R}^n \to [0, 1]$ such that

$$\int_{\mathbb{R}^n} f(x)\, dx = 1. \tag{1}$$

Such a function defines a probability measure

$$\Pr_\rho(A) = \int_A \rho(x)\, dx. \tag{2}$$

In dimension one, $\Pr([x, x + \Delta x])$ is the area under the curve between x and $x + \Delta x$:

$$\Pr([x, x + \Delta x]) = \int_x^{x+\Delta x} \rho(x)\, dx. \tag{3}$$

The Uniform Density The *uniform* density (with support on $[0, 1]$) is given by

$$\mathcal{U}(t) = 1. \tag{4}$$

So the probability of a random sample landing in one location is the same as the probability of landing in any other location.

$$\int_{x'}^{x'+\delta} dt = \int_x^{x+\delta} dt = \delta. \tag{5}$$

More generally, the uniform density on any domain V is given by the function

$$\mathcal{U}(x) = \begin{cases} \frac{1}{\mathrm{vol}(V)}, & x \in V \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

The Gaußian (Normal) Density The Gaußian density $N(\mu, \sigma^2)$ has support on $\mathbb{R}$ and is given by

$$N(\mu, \sigma^2)(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{7}$$

where μ is a real number and σ is a real number bigger than zero.

3.1 Random Sampling

A random sample *drawn from* a probability density ρ is characterized by a random uniform sample from the set

$$P = \{(x, u) \in V \times [0, 1] \,|\, u < \rho(x)\}. \tag{8}$$

This is often called the *Fundamental Theorem of Sampling* (FTS) (Robert and Casella, 2004). Given $(x, u) \in P$, the random sample x is said to be a *random variable* drawn from ρ. Random variables are traditionally denoted by capital letters, e.g. X and the notation

$$X \sim \rho$$

denoted that X is (a random variable) drawn from ρ.

Given an algorithm for sampling randomly and uniformly, the FTS leads to a method for sampling from any density ρ as will be seen shortly. First, a word or two is in order concerning random uniform sampling. This is in itself a complicated issue as the reader can see from Don Knuth's exposition in (Knuth, 1981). There is not enough room here to even summarize that exposition. It will be assumed that a good random uniform number is available. Most computer language compilers and systems have a built-in random uniform number generator so this is not an outrageous assumption. So given a method for drawing randomly and uniformly from any set, an algorithm for drawing randomly from a given probability density ρ is as follows.

Bound the graph of ρ by a rectangular box, sample it uniformly to find (x, u) and choose x if and only if $u < \rho(x)$. This is called the *accept-reject method* (ARM). Graphically, we visualize this as in Figure (2). Thus, given ρ, there is a way to find a random sample $X \sim \rho$.

3.2 Independence

In general, in dimensions $n > 1$ if $X \sim \rho$, then X is a vector $X = (X_1, \ldots, X_n)$ and $\rho(x_1, \ldots, x_n)$ is called the *joint probability density* of the random variables $X_1, \ldots, X_n$. If ρ is a tensor product as below

$$\rho(x_1, \ldots, x_n) = \prod_{i=1}^{n} \rho_1(x_1) \ldots \rho_n(x_n) \tag{9}$$

for some one dimensional densities ρ_i, $i = 1, \ldots, n$, the random variables $X_1, \ldots, X_n$ are said to be *independent*. A sequence of random variables $X_1, \ldots, X_n$ all

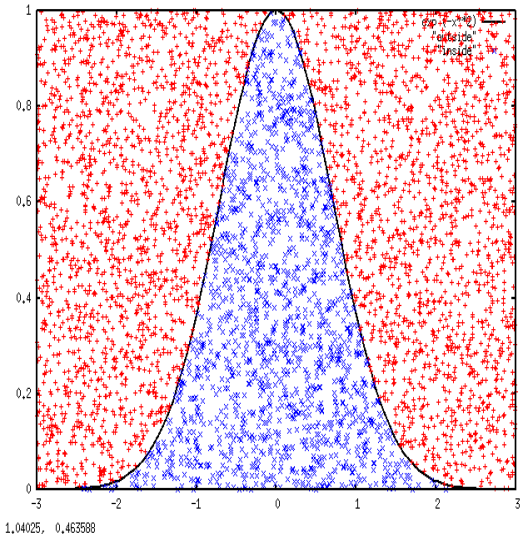

Figure 2: The ARM

drawn randomly from some ρ is said to be *independent and identically drawn* (IID) from ρ.

3.3 Functions of a Random Variable and Expectation

Given $X \sim \rho$ in dimension one, the *expectation* of X or *expected value*, or *mean* (with respect to ρ) is given by

$$E_\rho(X) = \int_V x\rho(x)\,dx. \tag{10}$$

If f is a function $f : \mathbb{R} \to \mathbb{R}$ then $Y = f(X)$ is also a random variable and is drawn from some density that will be denoted by $f_*(\rho)$. With some effort, $f_*(\rho)$ can be found explicitly, but it is not necessary to do so in terms of expectation since it can be shown that

$$E_{f_*(\rho)}(Y) = \int_V y f_*(\rho)\,dy = \int_V f(x)\rho(x)\,dx \tag{11}$$

(by "change of variables" essentially).

In any dimension $n \geq 1$, if $f : V \to \mathbb{R}$ is a real valued function, the expectation of f with respect to ρ is given by

$$E_\rho(f) = \int_V f(x)\rho(x)\,dx, \tag{12}$$

provided that the integral exists. The expectation $E_\rho(f)$ is often written $\mu_\rho(f)$ or just $\mu(f)$.

Note that for and functions f and g, and for any constant c, we have

$$E_\rho(cf + g) = cE(f) + E(g). \tag{13}$$

Note here that we omitted the subscript ρ from the right hand side as is often done when the context is clear.

3.3.1 Variance and Covariance and Correlation

Given $X \sim \rho$ and any function f, the *variance* of f (with respect to ρ) is given by

$$Var_\rho(f) = E_\rho((f - E_\rho(f))^2). \tag{14}$$

Note that this is equivalent algebraically to

$$Var_\rho(f) = E(f^2) - E(f)^2. \tag{15}$$

The standard deviation of f (with respect to ρ) is given by $\sqrt{Var(f)}$ and is often denoted by σ_f.

An easy exercise in algebra and elementary calculus gives that for any functions f and g and any constant c, we have

$$Var_\rho(cf) = c^2 Var(f) \tag{16}$$

and if f and g are independent,

$$Var_\rho(f + g) = Var(f) + Var(g). \tag{17}$$

Given random variables X, Y in dimension one 1 and joint density $\rho(x, y)$, for any real valued functions f and g, the covariance is given by

$$Cov_\rho(f, g) = E((f - E(f))\,(g - E(g))). \tag{18}$$

Note that this is algebraically equivalent to

$$Cov_\rho(f, g) = E(fg) - E(f)E(g). \tag{19}$$

For 1D IID X_1 and X_2 simple algebra shows that

$$Var(X_1 + X_2) = Var(X_1) + Var(X_2) + 2Cov(X_1, X_2). \tag{20}$$

Thus if X_1 and X_2 are independent, we have

$$Cov(X_1, X_2) = 0. \tag{21}$$

For two real valued functions of one real variable each, the correlation is given by

$$Cor_\rho(f, g) = \frac{Cov(f, g)}{\sqrt{Var(f)}\sqrt{Var(g)}}. \tag{22}$$

Extensions to higher dimensions is straightforward.

4 Statistical Modeling

4.1 Function Approximation

Consider a situation where a real valued function f defined on a subset of $\mathbb{R}^n$ is only partially known or is known, but is quite expensive to evaluate. Thus f is known on some subset $X = \{x_1, \ldots, x_m\}$ of the domain of f with values $Y = \{y_1, \ldots, y_m\}$. A goal is to use the values (X, Y) to somehow approximate f outside of X. The pair $(X.Y)$ is sometimes called a *training set*, X sometimes called the *design* (matrix), and Y is called the *response* (vector). This is a class of problems called *regression problems*. When the values of f land in a finite set and the domain is not necessarily $\mathbb{R}^n$, the class is called *classification problems* (Bishop, 2006). Here only regression problems are considered.

A Broad Outline A broad outline for solving regression problems is the following. Given a training set (X, Y) choose a (parameterized) *model family*:

$$\widehat{y}(x; \beta). \tag{23}$$

The function $\widehat{y}$ may be linear or non-linear. The $\beta = (\beta_1, \ldots, \beta_k)$ are called *model parameters*. When the model parameters are found, we have a model and the model is used to compute (predict) responses outside of the design space X.

How are the Best Model Parameters Determined? Intuitively, the training set can be used to obtain a function of β alone through the use of an *error function*:

$$\mathcal{E}(\beta) = \sum_{i=1}^{m} ((\widehat{y}(x_i; \beta) - y_i)^2 \tag{24}$$

which should be minimized. Specific forms for model families $\widehat{y}$ give rise to specific families of regression model types. Perhaps the most familiar are the linear least squares regressors and feedforward neural nets (Bishop, 1996). More will be said about this in Section 4.3 below.

4.2 Basis Functions

One way to build model families is to choose a particular space of functions $\mathcal{F}$ and a *basis family* for $\mathcal{F}$. A basis family for $\mathcal{F}$ is any set of linearly independent functions (over $\mathbb{R}$) $B = \{\phi_0, \phi_1, \ldots\}$ such that any function $f \in \mathcal{F}$ can be expressed as a linear combination

$$f = \sum_{i=0}^{p} \beta_i \phi_i \tag{25}$$

where p is a non-negative integer or infinity. In the cases to be considered here, p will always be less than infinity. For example, when $\mathcal{F}$ is the space of

polynomials in one variable t and of degree less than or equal to p, we have a basis family $B = \{\phi_0, \phi_1, \ldots, \phi_p\}$ where

$$\phi_i(t) = t^i. \tag{26}$$

This basis family is called the *monomial basis family*.

Tensor Product Bases Note that in Equation (25), the index i is allowed to be a multi-index, e.g. $(i_0, i_1, \ldots, i_p)$ where i_j is a non-negative integer. Generally, the *degree* of such a multi-index is given by

$$\deg(i_0, i_1, \ldots, i_p) = i_0 + \cdots + i_p. \tag{27}$$

For example, if $\phi_0, \phi_1, \ldots$ is a one dimensional basis family, we can form an n dimensional basis family by taking *tensor products* (where in the equation below, $x = (x_0, \ldots, x_{n-1})$)

$$\phi_{(i_0,i_1,\ldots,i_{n-1})}(x) = \phi_{i_0}(x_0)\phi_{i_1}(x_1)\ldots\phi_{i_{n-1}}(x_{n-1}). \tag{28}$$

For example, if $\mathcal{F}$ is the space of multivariate polynomials of total degree d, the tensor products of the monomial basis family give basis family

$$\{\phi_{(i_0,i_1,\ldots,i_{n-1})} \,|\, \deg(i_0, i_1, \ldots, i_{n-1}) \leq d\}$$

where

$$\phi_{(i_0,i_1,\ldots,i_{n-1})}(x) = x_0^{i_0} x_1^{i_1} \ldots x_{n-1}^{i_{n-1}}. \tag{29}$$

This basis family is called the *monomial basis family*.

Some Other Basis Families The Chebychev basis family (of the first kind) on $[-1, 1]$ is defined by

$$T_n(x) = \cos(n \cos^{-1}(x)). \tag{30}$$

These functions are *orthogonal* an $[-1, 1]$, i.e. the satisfy the equation

$$\int_{-1}^{1} T_m(x)T_n(x)\omega(x)\,dx = h_{i,j} \tag{31}$$

where $h_{i,j} = 0$ if $i \neq j$ and $h_{i,i} > 0$. In this particular case, we have

$$h_{i,i} = \begin{cases} \pi & i = 0 \\ \frac{\pi}{2} & i > 0 \end{cases} \tag{32}$$

and

$$\omega(x) = \frac{1}{\sqrt{1 - x^2}}. \tag{33}$$

There are many other useful orthogonal basis families such as the Jacobi, Legendre, Gagenbauer, Fourier, Hermite families. See (Szegő, 1975; Boyd, 2001) for more information.

The monomial and many orthogonal basis families have been implemented by the author in ANSI C as a part of the work done in (Lambe, 2011).

4.3 Some Parameterized Model Families

The model family for OLLSR (Section 5 below) is given by

$$\widehat{y}(x;\beta) = \sum_{i=0}^{p} \beta_i \phi_i \tag{34}$$

for some chosen p and basis family $\phi_0, \ldots, \phi_p$. Note that the basis functions themselves can be (and usually are) nonlinear, but the model is linear in the basis functions.

The model family for feed-forward neural nets is much more involved and non-linear, but is again of the form $\widehat{y}(x;\beta)$ for some parameters. In this case, the parameters are of two types called *weights* and *biases*. See (Bishop, 1996).

5 Ordinary Linear Least Squares Regression

In this case, the model family is of the form

$$\widehat{y}(x) = \sum_{i=0}^{p-1} \beta_i \phi_i(x) \tag{35}$$

as mentioned above.

5.1 The Classic Solution

The error minimization problem for Equation (24) in this case has an algebraic solution. Let $F_{i,j} = \phi_j(x_i)$. Then β is a critical point if and only if

$$F^T F \beta = F^T Y \tag{36}$$

where the superscript T denotes transpose. We obtain an argmin when $F^T F$ is invertible and then

$$\beta = (F^T F)^{-1} F^T Y. \tag{37}$$

We obtain this algebraic result in this case (as an easy exercise in calculus and algebra shows) because the error function (in this case) is just a quadratic function in β.

The β_i, $i = 0, \ldots, p$ are often called *regression coefficients* in the literature.

5.2 OLLSR via Likelihood

If we have a family of (parameterized) probability densities $\rho(x|\beta)$, the associated likelihood function is given by

$$L(\beta, x) = \rho(x|\beta). \tag{38}$$

Bayes theorem states that

$$\rho(\beta|x) = \frac{\rho(x|\beta)\rho(\beta)}{\rho(x)}. \tag{39}$$

The likelihood function is usually based on samples $x_1 \dots x_m$. We then form

$$L(\beta) = \rho(x_1, \dots, x_m|\beta). \tag{40}$$

As Bayes' theorem indicates, maximizing the likelihood gives the greatest probability for β given $x_1, \dots, x_m$ (Bishop, 2006).

We can convert the maximization of $\mathcal{L}(\beta)$ into a minimization problem by taking the negative log

$$\mathcal{E}(\beta) = -\ln \mathcal{L}(\beta). \tag{41}$$

$\mathcal{E}(\beta)$ is called the *error function*.

Assume that a training set (X, Y) is given and, for a fixed x, the density ρ of y is Gaußian with mean $\widehat{y}(x;\beta)$ (an assumption made by Gauß (Bishop, 2006)) and variance one, i.e.

$$\rho(y|x;\beta) = \frac{1}{(2\pi)^{\frac{1}{2}}}\, e^{-\frac{1}{2}(y-F(x;\beta))^2} \tag{42}$$

Assuming that the samples x_i are independent and identically distributed (IID), the associated likelihood function with respect to the given training set is

$$k \prod_{i=1}^{m} e^{-\frac{1}{2}(y_i - \widehat{y}(x_i;\beta))^2} \tag{43}$$

where the constant k is the appropriate constant. The associated error function is

$$\mathcal{E}(\beta) = c_1 + c_2 \sum_{i=1}^{m} (y_i - \widehat{y}(x;\beta))^2 \tag{44}$$

and this is exactly the intuitive error function of Equation (24)! Note that no assumption was made on the model family above, but if we assume that the model family is of the form

$$\widehat{y}(x;\beta) = \sum_{i=0}^{m-1} \beta_i \phi_i(x) \tag{45}$$

where $\phi_0, \dots, \phi_p$ is some sequence of basis functions then we obtain exactly OLLSR.

The OLLSR model has been implemented by the author in parallelized ANSI C as a part of the work done in (Lambe, 2011).

6 A General Procedure for Regression

Based on the observations in the previous Sections, we can formulate a general procedure for deriving regression models as follows. Given a training set (X, Y) which is believed to come from some function $y = f(x)$,

- choose a parameterized model family $\widehat{y}(x; \beta)$.
- derive an error function (sometimes called a *loss* function) that uses the training set and only depends on β,
- minimize the loss function to obtain the model, and
- derive a measure of goodness for the model off the design X.

The last item is a new part that will be covered below. In Section 5, this procedure, except for the last part, was carried out under the following assumptions and actions. It was assumed that the design was IID drawn from the Gaußian density and that the y_i have (sample) mean the model itself. Using maximal likelihood theory then, an error function was derived and it was seen to be the usual least squares error function which, in this case, has an algebraic solution. Many other models, both well-known and still yet to surface can be derived using the general procedure. Four notable examples are *ridge regression* (see below), sparse vector machines (Bishop, 2006), and neural nets (Bishop, 1996) as well as Gaußian process regression (GPR) described in Section 8.

Ridge Regression As an example of how one can vary the error function in the procedure above, ridge regression replaces the least squares error function

$$\mathcal{E}(\beta) = \sum_{i=1}^{m} (y_i - y(x_i; \beta))^2 \tag{46}$$

by

$$\widetilde{\mathcal{E}}(\beta) = \lambda \beta^T \beta + \sum_{i=1}^{m} (y_i - y(x_i; \beta))^2 \tag{47}$$

for some "bias parameter" $0 \leq \lambda < 1$. This new error function can be derived by the assumption that the model parameters are distributed normally. The details can be found in (Bishop, 2006, §1.2.5). Several choices for λ are given in (Clark and Troskie, 2006). If a linear model with a choice of basis functions is given then just as before, ordinary calculus and linear algebra can be used to derive the argmin solution to minimizing $\widetilde{\mathcal{E}}$. The solution is given by

$$\widehat{\beta} = (F^T F + \lambda^2 I)^{-1} F^T Y. \tag{48}$$

A positive effect of the new error function is that more insignificant regression coefficients tend to automatically have smaller values than in least squares regression.

7 A Goodness Measure for OLLSR

Given a model $\widehat{y}$ for a function y, *mean squared error* at x

$$\text{MSE}(x) = E_{f_*(\rho)}\left((\widehat{y}(x) - y(x))^2\right) \tag{49}$$

where $y \backsim f_*(\rho)$ (recall Section 3.3 for $f_*(\rho)$), an easy (algebraic) calculation gives

$$\begin{aligned} \text{MSE}(x) &= E(\widehat{y}^2) - E(\widehat{y})^2 + E(\widehat{y})^2 - 2yE(\widehat{y}) + y^2 \\ &= \text{Var}(\widehat{y}(x)) + \text{bias}(x)^2 \end{aligned}$$

where

$$\text{bias}(x) = E(\widehat{y}(x)) - y(x). \tag{50}$$

A model is *unbiased* if $\text{bias}(x) = 0$. Such is the case for OLLSR.

An important observation in the case of OLLSR is the following. Recall that the regression coefficients are given by (in matrix form)

$$\beta = (F^T F)^{-1} F^T Y \tag{51}$$

(Equation (37)). This means that if we express the model in matrix form, viz.

$$\widehat{y}(x) = \Phi^T(x)\beta = \Phi^T(x)(F^T F)^{-1} F^T Y = S(x)Y = \sum_{i=1}^{m} S_i(x) y_i \tag{52}$$

where

$$\Phi(x) = \begin{bmatrix} \phi_0(x) \\ \vdots \\ \phi_p(x) \end{bmatrix} \tag{53}$$

and

$$S = (F^T F)^{-1} F^T \tag{54}$$

(a one dimensional vector), we see that the model is a linear combination of random variables (the y_i) which are IID since it was assumed that the x_i were IID. The upshot of all of this is that we may compute the MSE as

$$MSE(\widehat{y}(x)) = Var\left(\sum_{i=1}^{m} S_i(x) y_i\right) = \sum_{i=1}^{m} S_i(x)^2 \sigma^2 = \sigma^2 ||S(x)||^2. \tag{55}$$

where σ^2 is the common variance of the random variables y_i (since they are IID). This is the *essence* of the heart of the seminal work on *confidence bands* by Scheffé in (Scheffé, 1999).

7.1 Confidence Bands for OLLSR

Let $\alpha \in (0,1)$ and assume that $\hat{r}(x)$ is some estimator. A $1-\alpha$ confidence band for $\hat{r}$ is a pair of functions (that in general depend on α) $a(x)$ and $b(x)$ such that

$$\Pr(a(x) \leq \hat{r}(x) \leq b(x)) = 1 - \alpha \tag{56}$$

From above, it is reasonable to assume that bounds for $\hat{y}$ should be of the form $\delta\sigma^2||S(x)||^2$, the problem is to determine δ in a way that, for a given α, a $1-\alpha$ confidence band results. In (Scheffé, 1999), this is done using α percentiles of the F-density, $F(p, m-p)$ where, in general, the F density is given by

$$F(a,b)(x) = \frac{\sqrt{\frac{ax)^a b^b}{(ax+b)^{a+b}}}}{xB(\frac{a}{2}, \frac{b}{2})} \tag{57}$$

and B is the beta function

$$B(c,d) = \frac{\Gamma(c)\Gamma(d)}{\Gamma(c+d)} \tag{58}$$

(Γ is the gamma function). This type of confidence band along with some others will be added to the libraries. Note that in practice, the term σ^2 above is approximated by an unbiased estimate of the variance (using the training set).

An Example Here is a very simple example. Consider the training set $\{(-1, b-a), (1, b+a)\}$ and model family $\beta_0 + \beta_1 x$. The F-matrix is

$$\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}. \tag{59}$$

We have that

$$F^T F = \frac{1}{2} I \tag{60}$$

so that

$$S(x) = [1, x](F^T F)^{-1} F Y = \frac{1}{2}[1, x] \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} = \frac{1}{2}[1-x, 1+x] \tag{61}$$

and so

$$||S(x)||^2 = \frac{1}{4}(1 - x^2). \tag{62}$$

A corresponding confidence band is given in Figure (3).

Note that there is an exact fit at both endpoints, i.e. on the training set, but as we move away from either endpoint, there is less confidence in the accuracy of the model. This shows that this kind of band can be overly cautious in reporting confidence; however only two training points were given! If one adds a third halfway between the endpoints, a much better band results.

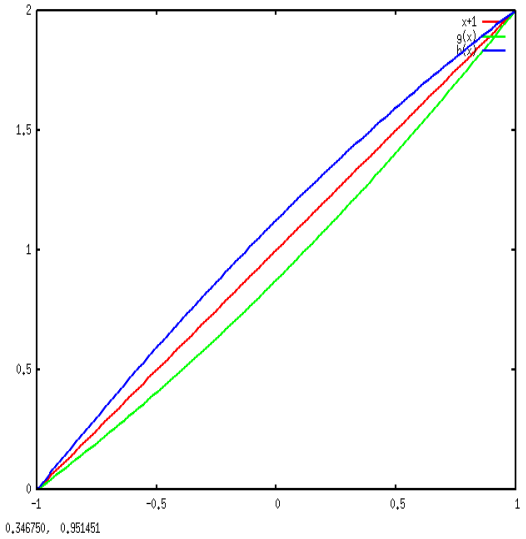

Figure 3: A plot of $x + 1 - \frac{1}{4}||L(x)||^2$, $x + 1$, and $x + 1 + \frac{1}{4}||L(x)||^2$

8 Gaußian Process Regression (GRP)

As another example of the general procedure for regression given in Section 6, we consider another model family that has received quite a bit of attention in the engineering community since 1989 (Sacks et al., 1989). Again, one also needs a training set and and an error function but this time a choice of correlation matrix is needed. The correlation matrix can be *any* symmetric positive definite matrix, but a correlation matrix based on the tensor product of n one-dimensional Gaußian functions see Equation (66) is seen quite a bit in the engineering literature. The correlation matrix R is actually of the form

$$R_{i,j} = C(x_i - x_j) \tag{63}$$

for some "correlation function" C. The model family goes back to (Krige, 1966) and is of the form

$$\Phi(x)^T\beta + z(x) \tag{64}$$

where the β here are *generalized* regression coefficients (with respect to R).

The model takes on the form

$$\Phi(x)^t\beta + r(x)^t R^{-1}(Y - F\beta) \tag{65}$$

where $\bar{r}(\bar{x})^t = (C(\bar{x} - \bar{x}_0), \ldots, C(\bar{x} - \bar{x}_{m-1}))$. The significance of the second term is that it turns the (generalized) regression term *into an interpolation*! The problem is that we *do not* actually know the true correlation function. A

solution to this dilemma uses a fairly standard mathematical trick, viz. choose a parameterized family $C_{\bar{\theta}}$ of correlation functions, e.g. tensor products of

$$C_\theta(x) = e^{-\left(\frac{x}{\theta}\right)^2} \tag{66}$$

and set up an optimization problem to find optimal values of the parameters θ_i. In this case, the optimization problem is once again derived using maximal likelihood theory.

So in GPR, the model family is different from that of OLLSR. We have to take a (parameterized) *correlation* matrix R_θ into account and the model family looks like

$$\sum_{i=0}^{p} \beta_i \phi_i(x) + \sum_{i=1}^{m} \gamma_i r_i(x) \tag{67}$$

where r_θ is related to R_θ. As with OLLSR, the model is unbiased and the mean square error at x can be derived. The formulas however are much more "gruesome"!

The GPR model has been implemented by the author in parallelized ANSI C as a part of the work done in (Lambe, 2011).

This lecture ends with two examples. The first comes out of work with Ben Grier (Clemson) (Grier, 2011) and shows that one needs to choose the correlation function correctly to ensure a proper GPR model and the second comes out of the work (Kolonay and Lambe, 2012) and shows how the $MSE(x)$ can be used to adaptively derive an optimal design.

8.1 Choosing the Correlation Function Correctly

Consider the piecewise function

$$y = \begin{cases} x^2 - \frac{1}{2}x + \frac{17}{16}, & 0 < x < \frac{1}{3} \\ \\ -17.875x^2 + 17.854x - 2.9583, & \frac{1}{3} < x < \frac{2}{3}. \end{cases} \tag{68}$$

Ordinary linear least squares regression with basis functions $\{1, x\}$ was performed piecewise on the *two intervals separately*. The result shown in Figure (4) used a 15 point training series and was, as expected, quite bad. By switching to GPR, the answer becomes much better despite using the same monomial basis function of degree one. One of the reasons for this is that GPR is an *interpolation* on the training set. Again, however, the two pieces were modeled separately. The result is shown in Figure (5). To test how good GPR is globally, a GPR model without incorporating the knowledge that there are two separate curves was tried. The Gaußian correlation function was used. The result is in Figure (6). The problem, of course, is that the GPR model with Gaußian correlation function is actually an *analytic* model and our true function is far from analytic (globally). The correlation was switched to one based on cubic splines and the model was corrected. The result is in Figure (7).

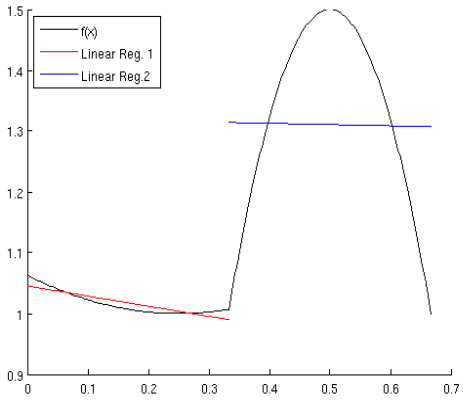

Figure 4: The OLLSR – linear basis, 15 points

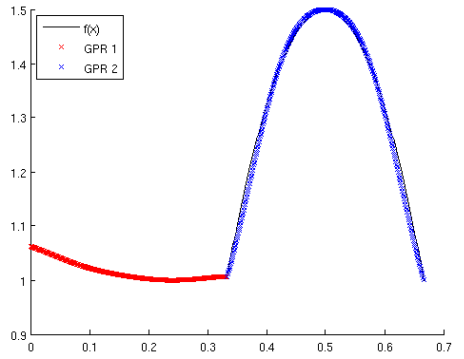

Figure 5: GPR using five points and a Gauss correlation function (linear basis)

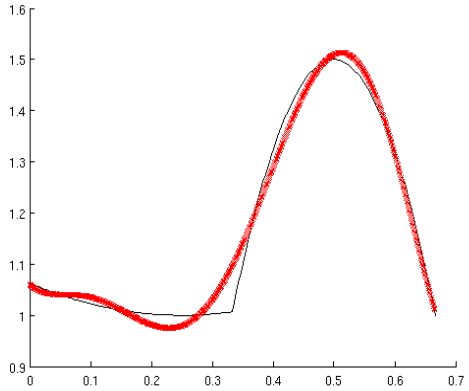

Figure 6: GPR using 20 points globally and a Gauss correlation function (linear basis)

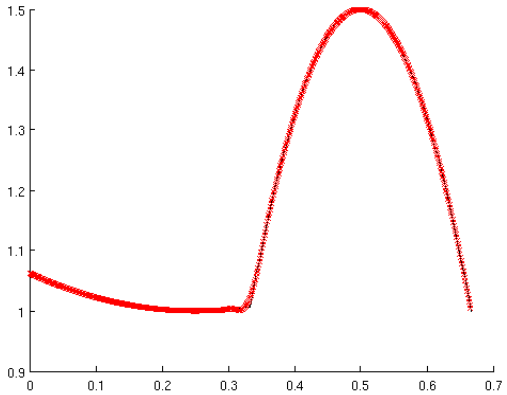

Figure 7: GPR using 40 points and a cubic correlation function

8.2 An Adaptive Method for Optimal Designs

In (Kolonay and Lambe, 2012), the goal is to find a surrogate for a black box objective function which is expensive to evaluate. The surrogate is a Gaußian process regression (GPR) model. We want to find the best model possible with the fewest number of training points. A recursive procedure was developed to adaptively grow a small training set in an optimal way.

Suppose that some statistical model $\widehat{y}(x)$ has the property that there exists some *measure of goodness* $S(x)$ of the model at a given point x. Consider the following algorithm starting with a training set (X, Y) where the domain of interest is V.

0: Use (X, Y) to derive a new model.

1: If CRIT return the model.

2: Let $W \subseteq V - X$.

3: Let $W' = \{w \in W \,|\, S(w) > TOL\}$

4: Compute responses Y' from W'.

5: Let $X = X \cup W'$ and $Y = Y \cup Y'$.

6: goto 0:

Runge's Example from the GPR Viewpoint In 1901, Carl Runge gave an interesting example that showed a problem with polynomial interpolation. *If the grid is taken to be uniform*, the model is quite bad. The example is simply

$$f(x) = \frac{1}{\sqrt{1 + 25x^2}}, \text{ on } [-1.1]. \tag{69}$$

He found that if the design (grid) is drawn from the Chebychev density

$$\rho(x) = \frac{1}{\pi} \frac{1}{\sqrt{1 - x^2}} \tag{70}$$

the interpolation was quite good even in high degrees.

The purpose of this exercise is to answer the following two questions.

1. What might this example look like using the GPR model?
2. What is the effect of the adaptive algorithm on a small initial training set?

One measure of goodness of the GPR model is given by the $S(x) = MSE(x)$. An important property of S in the case of the GPR model is that

$$x \in X \text{ implies } S(x) = 0. \tag{71}$$

We chose an initial uniform design with 11 points, viz.

$$X = \{-1, -0.8, -0.6, \ldots, 0, \ldots, 0.6, 0.8, 1\} \tag{72}$$

and the exact function was used to generate the responses Y. The Legendre family was chosen and used up to and including degree ten. A refinement X' of the initial design was given (so W is the complement $W = X' - X$). The function S was evaluated on X'. The numerical results follow. The output: first column x, second column S(x):

```
-1.000000 0.000001
-0.900000 13800.554459
-0.800000 0.000000
-0.700000 617.908453
-0.600000 0.000000
-0.500000 66.156155
-0.400000 0.000000
-0.300000 9.642134
-0.200000 0.000000
-0.100000 0.705735
-0.000000 0.000000
0.100000 0.705735
0.200000 0.000000
0.300000 9.642134
0.400000 0.000000
0.500000 66.156155
0.600000 0.000000
0.700000 617.908453
0.800000 0.000000
0.900000 13800.554459
1.000000 0.000001
```

The graph of this first model is given in Figure (8). A tolerance of $TOL = 10$ was used for the adaptive method, i.e. $S(x) > 10$ responses were computed using the exact function and the model was recomputed. The graph of the new model is in Figure (9). It was found that that after more iterations, most bad points were concentrated towards the endpoints. This coincides with Runge's observations!

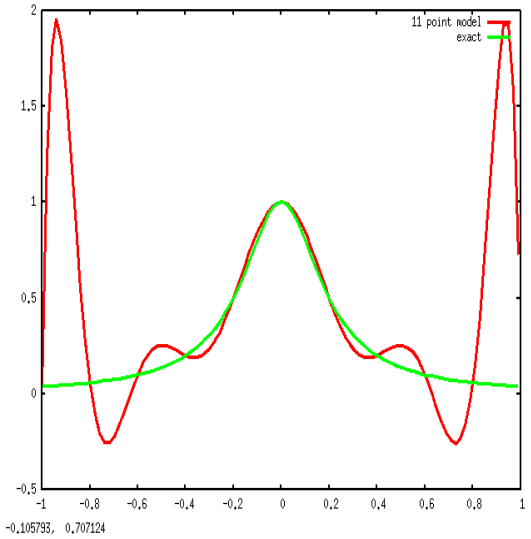

Figure 8

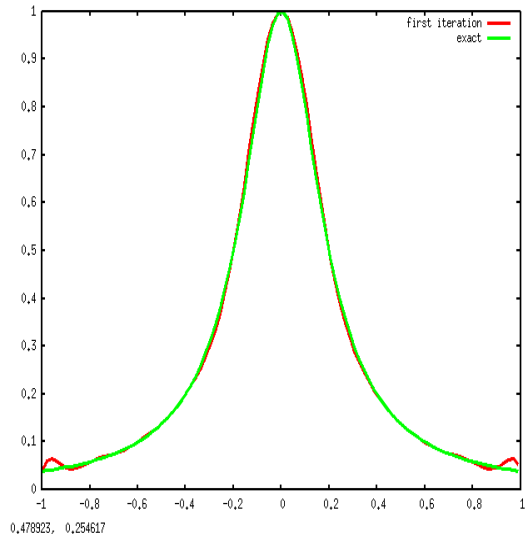

Figure 9

References

Bishop, C. M. (1996). *Neural Networks for Pattern Recognition*. Oxford University Press.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Information Science and Statistics. Springer, New York.

Boyd, J. P. (2001). *Chebyshev and Fourier spectral methods*. Dover Publications Inc., Mineola, NY, second edition.

Clark, A. E. and Troskie, C. G. (2006). Ridge regression—a simulation study. *Comm. Statist. Simulation Comput.*, 35(3):605–619.

Gauss, C. F. and Davis, C. H. (1857). *Theory of the motion of the heavenly bodies moving about the sun in conic sections : a translation of Gauss's "Theoria motus" / with an appendix by Charles Henry Davis*. Little, Brown, Boston.

Grier, B. (2011). AFRL Summer visitor.

Hald, A. (2007). *A History of Parametric Statistical Inference From Bernoulli to Fisher, 1713-1935*. Springer-Verlag, New York.

Harlow, F. H. and Metropolis, N. (1983). Weapons simulation leads to the computer era. *Los Alamos Science*, 7. http://la-science.lanl.gov/cat_history.shtm.

Kalos, M. H. and Whitlock, P. A. (2008). *Monte Carlo methods*. Wiley-Blackwell, Weinheim, second edition.

Knuth, D. E. (1981). *The art of computer programming. Vol. 2*. Addison-Wesley Publishing Co., Reading, Mass., second edition. Seminumerical algorithms, Addison-Wesley Series in Computer Science and Information Processing.

Kolonay, R. M. and Lambe, L. A. (2012). Experiments with Gaußian process regression for a problem in wing design.

Krige, D. G. (1966). A study of gold and uranium distribution patterns in the klerksdorp goldfield. *Geoexploration*, 4:43–53.

Lambe, L. A. (2006–2011). Air Force (AFRL, WPAFB) SBIR, Phases I and II. Contract No. FA8650-06-C-3621.

Robert, C. P. and Casella, G. (2004). *Monte Carlo statistical methods*. Springer Texts in Statistics. Springer-Verlag, New York, second edition.

Rubinstein, R. Y. and Kroese, D. P. (2008). *Simulation and the Monte Carlo method*. Wiley Series in Probability and Statistics. Wiley-Interscience [John Wiley & Sons], Hoboken, NJ, second edition.

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989). Design and analysis of computer experiments. *Statist. Sci.*, 4(4):409–435. With comments and a rejoinder by the authors.

Scheffé, H. (1999). *The analysis of variance*. Wiley Classics Library. John Wiley & Sons Inc., New York. Reprint of the 1959 original, A Wiley Publication in Mathematical Statistics.

Szegő, G. (1975). *Orthogonal polynomials*. American Mathematical Society, Providence, R.I., fourth edition. American Mathematical Society, Colloquium Publications, Vol. XXIII.