"Looking For Nothing" Bayes Linear Methods for Solving Equations.

A thesis presented for the
degree of Doctor of Philosophy
at the University of Durham.

James A. Smith[‡]

Department of Mathematical Sciences,
University of Durham,
Durham, DH1 3LE.
ENGLAND.

January 1993

[‡]Supported by a UK Science and Education Reseach Council research studentship.

Abstract

Here I will describe and implement Bayes linear methods for finding zeros of deterministic functions. We assume that the zero is known to be unique. Initially, the value of the function is modelled simply as the product of two independent factors, the position of the point from the zero and a "slope" which is assumed to vary "smoothly" with position. Additional prior information specifies first and second order properties of the slopes and the position of the zero: in particular, smoothness is specified by modelling the slope process to be stationary with a decreasing correlation function.

This research is motivated by problems arising in large scale computer simulation of mathematical models of complex physical phenomena, where a single run of the code can be expensive and the output difficult to assimilate. Scientists are often confident about the structure of their model as a description of a physical process but may be uncertain about the values of certain model "parameters". Such parameters usually refer directly to physical attributes, and so collateral information about their values is usually available. In some applications, the physical process itself has been observed, and several runs of the code are made at different parameter settings in an attempt to match the realisation of the code with the actual realisation.

The eventual aim is to aid scientists to search through the "parameter space" efficiently and systematically, using their knowledge of the process. Obviously, there are several respects in which this formulation does not tackle the real problem, as we mainly consider a single-valued function of a real variable.

As well as considering this problem I will review the current state of play in the more general field of statistical numerical analysis and its relationship to deterministic computer experiments; and partial belief specification or Bayes linear methods.

Dedication

I dedicate this to my parents, for the support they have given me, especially over the last six years as a student in Durham; to my brother who has had to put up with me for twenty-four years; and to the Cub Scouts of Durham City and County, without whom Chapter 4 may never have seen the light of day.

Acknowledgements

I would like to thank: my supervisor, Allan Seheult, for the contributions he has made to the problem, and for his reading of the first and final drafts of the document; Michael Goldstein for numerous discussions on his notion of partial belief adjustment; and to Professor O'Hagan for the discussion we had at the Fourth International Meeting of Bayesian Statistics In Peñiscola, Spain.

As well I would like to thank the SERC for the grant which has paid my way for the last three years, and who will be paying my wages for the next three.

Statement of Originality

The work of Chapters 4 and 5 and Appendices B and C are all my own work, along with the examples in Chapter 3 and Appendix A, and some of the calculations in Sections 2.2.1 and 2.2.2.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Motivation

With the advent of more powerful computers, scientists have been able to develop and try out more and more complicated mathematical models of many physical processes, often when physical experimentation is impossible (for example in modelling the earth's atmosphere for weather forecasting) or very expensive in either time or money or both.

Examples of these models can be seen in many branches of science and industry, meteorology (weather forecasting), physics (transmission of heat), chemistry (reaction kinetics), computing (design of VLSI silicon chips), surveying (underground surveys by remote sensing).

These models have the following characteristics,

1 They generally consist of large systems of non-linear equations, with multi-dimensional inputs and outputs;

2 They are deterministic, i.e. the same set of input values will always produce the same response;

3 A single run of the code is expensive, some examples take several minutes to run on modern super computers (e.g. Cray X-MP).

They have many uses which fall into three categories,

1 Prediction/Forecasting – obtaining an approximation for the function at untried input values, either in the design space 'prediction' or outside 'forecasting';

2 Calibration – fitting the computer code to the actual observed data;

3 Optimization – finding input values which maximize or minimize the response function or some functional of it;

In this thesis we will be mainly considering a simple version of the second category, where the input is univariate.

These (deterministic) problems are akin to those that numerical analysts tackle: approximation, solving equations, and optimization. The methods developed for solving these can also be used to solve the numerical analysts problems. The numerical analyst's functions are often much cheaper to compute than the results of the deterministic simulations, and so they do not have as tight a constraint on the number of observations they can take. We will expand on this in Chapter 2, which contains a review of the current work on the subject.

As there are no random error terms in these models, it is not clear how a statistical approach can help solve these problems. There is however a large number of unknowns (the function value at every point we have not made an "observation", i.e. function evaluation, at), which are "correlated" to each other. We can consider the approximation of the function or its attributes as a problem of experimental design – we need an estimate of the function or its attribute with a measure of uncertainty, and we need to choose the input values of the computer code in such a way as to minimize the number of times we have to run the program.

We choose a series of design points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, at which we can obtain observations y_i (by running the computer code with $\boldsymbol{x}_i$ as the input variables), obtain an estimate of the function at other values of $\boldsymbol{x}$ and quantify the uncertainty of these predictions. From a statistical viewpoint the Bayesian paradigm lends itself very naturally to this problem, again we will expend on this point in Chapter 2, where other more classical approaches require contrived reasoning. The main problem with Bayesian statistics, when applied to real problems is that the calculation of posterior distributions is often computationally difficult involving multi-dimensional integrals of multi-modal functions. In an attempt to avoid some of these problems we will utilize the Bayes Linear

framework of Goldstein[1981, 1983, 1986, 1987, 1988a, 1988b, 1991], which is summarised in Chapter 3, and we will introduce a new design criterion for deterministic problems based on this in Chapter 4

In Chapter 5 we will outline the methods and models developed to locate zeros of functions, initially where the input and output are both univariate, but extending the model to include multivariate response and input.

Finally in this introductory chapter we will include a brief description of examples of the problems which have motivated this field of research

Example 1: Kee, Grcar, Smooke and Miller [1985] describe a fluid-dynamics model for flames, which contains five unknown parameters (rate constants), with a single response, the velocity of the flame. Their aim is to tune the model parameters to match physically observed results, and is an example where an equivalent physical experiment is impossible because the model parameters are fixed physical constants. The code for this example requires twenty minutes per run (each choice of parameters) on a Cray X-MP, so highlights the importance of the design problem of minimizing the number of runs to be performed.

Example 2: TWOLAYER is a simulator of heat transfer through a wall containing two layers (hence the name) of phase change materials. Output is a measure of the storage capacity of the wall and the aim is to maximize this capacity.

Example 3: FABRICS II [1984] is a simulator for silicon circuitry. Inputs are circuit parameters and production process parameters (some of which are allowed to vary) and the response is a measure of the circuit (for example, delay time). The aim is to choose values of design parameters so that the response is insensitive to variation in the process parameters.

Example 4: History matching - Reservoir engineers have developed models for the flow of fluids through the earth, which can be used to model oil and gas reservoirs. These models need the underground structure of the reservoir (geological faults, properties of the rock e.g. porosity and permiability - spatially distributed throughout the reservoir, etc.) as input parameters, and outputs are

typically time series, such as the flow/pressure of gas/oil/water at various well sites. Their aim is to match the output of the model to the actual observations at the well sites, and so predict the underground structure of the reservoir.

Chapter 2

Statistical Numerical Analysis

Most problems in numerical analysis entail approximating some facet of a given function, whether it is its value over a given interval, the location of its extremes, its integral or some other functional, which cannot be directly calculated, or whose direct calculation is expensive. In all these cases the numerical analyst evaluates the function (or a related function) at a set of ordinates, and uses these values to make his approximation. Statisticians can view these as problems of inference. We have a parameter, the desired functional which we want to approximate, that we wish to make an inference about, and we have some 'experimental observations', the function values on which to base this inference.

For example, suppose we wished to approximate the integral of a function $y(\cdot)$,

$$I = \int_{-1}^{1} y(x)dx$$

A numerical analyst would chose a quadrature rule comprising a set of ordinates $X = \{x_1, \ldots, x_n\} \subset [-1, 1]$ and a set of weights $W = \{w_1, \ldots, w_n\}$ and compute Q his approximation as

$$Q = \sum_{i=1}^{n} w_i y(x_i).$$

As a statistician we choose our design X for our 'experiment'; statistically model the function; make our 'observations' $\{y(x_1), \ldots, y(x_n)\}$; and use this to make inference

about I.

The numerical analyst's approach usually neglects any additional information we might have about the function, e.g. convexity and symmetry – when he chooses the set X, and his optimal design is based generally on *worse case* criteria, often choosing to fit for polynomials up to and including degree n. From a statistical point of view this *apriori* information can be incorporated into the design of the experiment, for example in a Bayesian framework, we can lay down a prior distribution on the functions which represents our beliefs about their shape.

This chapter contains a review of work on Bayesian numerical analysis, and more general statistical numerical analysis. We will generally consider those methods where the function evaluations are assumed to be error free, and so the approximant is interpolatory, but we will briefly consider the case where function evaluations are subject to error, which lead to numerical 'smoothers'. We start by considering deterministic computer experiments.

2.1 Deterministic computer experiments

Although this chapter is entitled *Statistical Numerical Analysis* it could equally have been entitled *Deterministic Computer Experiments*, as most problems in numerical analysis can be considered as deterministic computer experiments and vice-versa.

However, an important consideration is the cost of obtaining a single evaluation of our function. Whenever we are considering the sort of experiments outlined in the introductory chapter, the cost of a single observation is likely to be high, whereas function evaluations in a numerical analysis problem are often relatively cheap. Therefore, when solving deterministic computer experiment problems we can apply a more sophisticated technique (one which uses a more complicated procedure for choosing design points) to evaluate our functional, because each observation saved saves a lot more time and money. It has to be remembered that whichever method we use the cost of using it should not exceed the savings we make by reducing the number of

function evaluations.

Another consideration is the precision or accuracy of the function. Generally the precision in the numerical analyst's problems will be much greater (as the functions are simpler to compute) than in the computer experiment problems. In the latter there is a much greater chance of numerical rounding errors, due to the complexity of the code, and also model inaccuracy – as the code is modelling real-world phenomena, and even with very precise models, there is likely to be some simplification.

2.2 Interpolation

Perhaps one of the most common of all numerical problems is that of finding an approximation to a function $y(\cdot)$, given function evaluations $y_1, \ldots, y_n$ at $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$. Usually we would like the approximant to be an interpolant, i.e. match the function at the $\boldsymbol{x}_i$s. Non-interpolatory approximants will covered in Section 2.5.

I will now outline three statistical approaches to the interpolation problem. The first approach entails putting a prior distribution on functions by placing a prior distribution on the coefficients of its power series expression, the other two model the function explicitly as a realization of a random process.

2.2.1 Poincaré's approach

Probably the first example of statistical numerical analysis can be found in Poincaré's *Calcul de Probability*[1896]. In lesson 21 Poincaré examines the problem of approximating a function by a finite polynomial. In lesson 22, however, he expands this to infinite polynomials. He supposes $y(x)$ has a power series expansion $y(x) = A_0 + A_1 x + \ldots$, where the A_is are unknown. He then places a prior distribution on the A_is by supposing them to be independent, with zero mean, Gaussian random variables with variance σ_i^2. Then given function evaluations $y_1, \ldots, y_n$ at $x_1, \ldots, x_n$ he finds the expected values of the A_is.

Writing

$$g(x) = \sigma_0^2 + \sigma_1^2 x + \sigma_2^2 x^2 + \cdots,$$

Poincaré shows that the posterior mean of $y(x)$ given function evaluations $y_1, \ldots, y_n$ at $x_1, \ldots, x_n$ is given by

$$\hat{y}(x) = k_1 g(x_1 x) + \cdots + k_n g(x_n x)$$

where the k_is are given by

$$\begin{bmatrix} k_1 \\ \vdots \\ k_n \end{bmatrix} = \begin{bmatrix} g(x_1x_1) & \cdots & g(x_1x_n) \\ \vdots & \ddots & \vdots \\ g(x_nx_1) & \cdots & g(x_nx_n) \end{bmatrix}^{-1} \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

For example, if we believe that the standard deviation of A_i is half the standard deviation of A_{i-1}, this gives us $\sigma_i^2 = \frac{\sigma_0^2}{4^i}$ and $g(x) = (1 - \frac{x}{4})^{-1}$ for $x \in (-4, 4)$. If we choose the design $x_1 = -1, x_2 = 0, x_3 = 1$, we have

$$\hat{y}(x) = \frac{-32y_2 + 15(y_1 - y_3)x + (32y_2 - 15y_1 - 15y_3)x^2}{2x^2 - 32}$$

2.2.2 Brownian motion

Diaconis[1988] looks to Brownian motion as a way to model the function. A particle exhibits Brownian motion if its velocity is always changing, defined as

Defn. 2.1 Brownian Motion *with drift parameter μ and variance parameter $\sigma^2 > 0$ is a continuous state and time stochastic process $Y(x)$ which exhibits the following properties.*

$$(i) \quad Y(x_1) - Y(x_0) \sim \mathbb{N}(\mu(x_1 - x_0), \sigma^2(x_1 - x_0))$$

$$(ii) \quad Y(x_1) - Y(x_0) \text{ and } Y(x_3) - Y(x_2) \text{ are independent}$$

$$\text{if } x_3 > x_2 \geq x_1 > x_0$$

Standard Brownian motion *has* $\mu = 0$ *and* $\sigma^2 = 1$.

The path of the particle is a continuous, but nowhere differentiable function of x, and exhibits the strong Markov property, i.e. knowing where the particle is at time x is all that is sufficient to predict its future position.

We can now identify our function as a possible path of the particle, and if we additionally specify a prior distribution for $Y(0)$, then we have a full prior distribution of paths, or functions in $C(-\infty, +\infty)$.

Let us consider functions on the unit interval $[0, 1]$, with evaluations at $y_1, \ldots, y_n$ at $0 \leq x_1 < \cdots < x_n \leq 1$, with a standard Brownian motion prior distribution, and with a normal improper ignorance prior distribution on $Y(0)$ $[\pi_{Y(0)}(y) \propto 1 \quad \forall y]$. As Brownian motion exhibits the Markov property the posterior mean of $Y(x)$ given $y_1, \ldots, y_n$ is just the linear spline interpolant.

$$\mathrm{P}(Y(x)|y_1, \ldots, y_n) = \begin{cases} y_1 & \text{if } 0 \leq x \leq x_1 \\ \frac{y_i(x_{i+1}-x)+y_{i+1}(x-x_i)}{x_{i+1}-x_i} & \text{if } x_i < x \leq x_{i+1} \\ y_n & \text{if } x_n < x \leq 1 \end{cases}$$

$$\mathrm{Cov}(Y(x), Y(x_*)|y_1, \ldots, y_n) = \begin{cases} x_1 - \max(x, x_*) & \text{if } 0 \leq x, x_* \leq x_1 \\ \frac{(x_{i+1}-\max(x,x_*))(\min(x,x_*)-x_i)}{x_{i+1}-x_i} & \text{if } x_i < x, x_* \leq x_{i+1} \\ \min(x, x_*) - x_n & \text{if } x_n < x, x_* \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

The simple posterior mean and covariance structures yield straight forward optimal designs for the interpolant based on the variance of the predictor, if we can chose n points then we have the following optimal designs:

Minimum Maximum Variance $\{\frac{4i-3}{4n-2}, i = 1, \ldots, n\}$

Minimum Average Variance $\{\frac{3i-2}{3n-1}, i = 1, \ldots, n\}$

(Note: these results are not those obtained in Diaconis [1988])

If we integrate the interpolant, and choose a design which minimizes the variance of the integral, we obtain the composite mid-point rule, with design ordinates $\{\frac{2i-1}{2n}, i =$

$1, \ldots, n\}$, and

$$\int_0^1 y(x)dx \simeq \frac{1}{n}\sum_{i=1}^{n} y(\frac{2i-1}{2n})$$

If we use integrated Brownian motion as our prior distribution, then the posterior expectation is the interpolatory cubic spline. Indeed if we repeat this integration m times we obtain splines of order $2m+1$ as our posterior estimate. See Wahba[1978]

2.2.3 Stationary stochastic processes

The approach employed by Sacks *et al*[1989a, 1989b], Schagen[1979, 1980a, 1980b] and Currin *et al*[1991] is also to model the function as a realization of a stochastic process. In the two papers by Sacks *et al*[1989a, 1989b] the response is modelled by a linear regression term plus an additional "error" term $Z(\cdot)$ with mean zero,

$$Y(\boldsymbol{x}) = \sum_{i=1}^{m} f_i(\boldsymbol{x})\beta_i + Z(\boldsymbol{x}) \tag{2.1}$$

where the $f_i(\cdot)$s are known functions and β_is are unknown regression parameters, and $\boldsymbol{x} \in \mathbb{R}^q$. This additional term is assumed to be a realisation of a stationary stochastic process, with mean zero, fixed variance σ^2, and the covariance between two observations a continuous function solely of their relative positions (not their absolute positions),

$$\mathrm{Cov}(Z(\boldsymbol{x}), Z(\boldsymbol{x}^*)) = \sigma^2\rho(\boldsymbol{x} - \boldsymbol{x}^*) \tag{2.2}$$

They assume also that the $Z(\cdot)$s have a multivariate normal distribution, although this is not necessary to obtain linear estimates.

Schagen[1979, 1980a, 1980b] and Currin *et al*[1991] consider the special case of (2.1) when

$$Y(\boldsymbol{x}) = \mu + Z(\boldsymbol{x}) \tag{2.3}$$

with $Z(\cdot)$ as before.

In one dimension, some of the correlation functions considered by these authors are (for functions defined on any interval of the whole real line, which are plotted in

Figure 2.1)

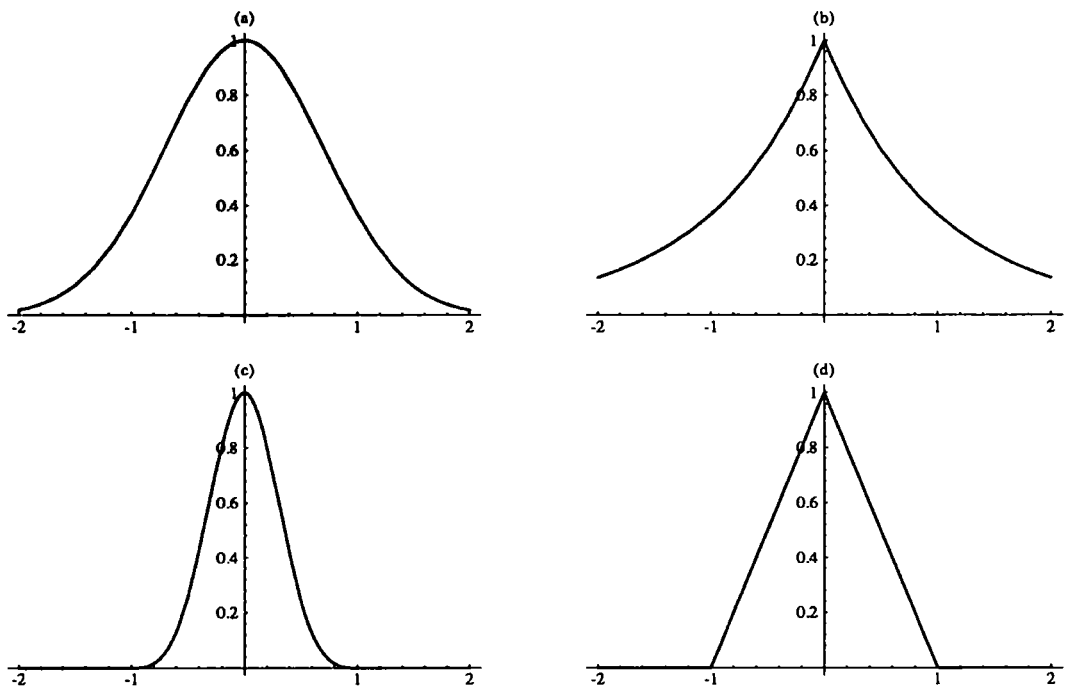

Figure 2.1: Correlation functions for (a) $\rho_2(\cdot)$, (b) $\rho_1(\cdot)$, (c) $\rho_{c+}(\cdot)$ and (d) $\rho_{l+}(\cdot)$, with $\theta = 1$

$$
\begin{aligned}
\rho_p(d) &= e^{-\theta|d|^p} && && \theta \geq 0, p > 0 \\
\rho_{l+}(d) &= \begin{cases} 1-\frac{|d|}{\theta} & |d| < \theta \\ 0 & \text{otherwise} \end{cases} && && \theta \geq 0 \\
\rho_{c+}(d) &= \begin{cases} 1-6\left(\frac{|d|}{\theta}\right)^2\left(1-\frac{|d|}{\theta}\right) & |d| < \frac{\theta}{2} \\ 2\left(1-\frac{|d|}{\theta}\right)^3 & \frac{\theta}{2} \leq |d| < \theta \\ 0 & \text{otherwise} \end{cases} && && \theta \geq 0
\end{aligned}
\tag{2.4}
$$

and for the interval $[0, 1]$

$$
\begin{aligned}
\rho_l(d) &= 1 - \frac{|d|}{\theta} && && \tfrac{1}{2} < \theta < \infty \\
\rho_c(d) &= 1 - \frac{\theta d^2}{2} + \frac{\theta'|d|^3}{6} && && \theta' \leq 2\theta \text{ and } \theta'^2 - 6\theta\theta' + 12\theta^2 \leq 24\theta'
\end{aligned}
\tag{2.5}
$$

In all of these θ controls the "smoothness" of the predictor, and so the range of

influence an observation has on the predictor.

Schagen[1979, 1980a, 1980b, 1984] uses $\rho_2(\cdot)$; Sacks *et al*[1989a, 1989b] use $\rho_p(\cdot)$ (in particular $\rho_2(\cdot)$) and also mention $\rho_c(\cdot)$ and $\rho_l(\cdot)$; and Currin *et al*[1991] use all of these. These functions have different properties. Only $\rho_2(\cdot)$ is continuously differentiable everywhere, where as $\rho_p(\cdot)$ (for $p \notin \{2, 4, \ldots\}$), $\rho_l(\cdot)$ and $\rho_{l+}(\cdot)$ are not differentiable at zero, and $\rho_c(\cdot)$ and $\rho_{c+}(\cdot)$ are twice differentiable everywhere. Both $\rho_l(\cdot)$ and $\rho_{l+}(\cdot)$ lead to piecewise linear interpolants and $\rho_c(\cdot)$ and $\rho_{c+}(\cdot)$ give cubic spline interpolants.

For simplicity it is often assumed that the correlation function $\rho(\boldsymbol{d})$ can be factorised into the product of q terms each dependent on only one of the co-ordinates of $\boldsymbol{d}$,

$$\rho(\boldsymbol{d}) = \prod_{i=1}^{q} \rho^{(i)}(d_i). \tag{2.6}$$

This assumption makes certain integrations easier to perform, as multiple integrals factorise into products of single integrals. However it does rule out certain correlation function that we might like to consider, for example $e^{-\theta\|\boldsymbol{d}\|}$.

It can be assumed that the correlation function is isotropic, i.e. the correlation function in each direction is of the same form with the same smoothness parameter. In practice though $\rho(\cdot)$ will not be isotropic but will be of the form

$$\rho(\boldsymbol{d}) = e^{-\sum_{i=1}^{q} \theta_i d_i^2} \tag{2.7}$$

Schagen[1980b, 1984] in his later papers adopts a 'half-way' approach by assuming he knows the anisotropy factors, and then "re-scaling".

$$\rho(\boldsymbol{d}) = e^{-\theta \sum_{i=1}^{q} \alpha_i d_i^2} \tag{2.8}$$

We will now discuss Schagen[1979, 1980a, 1980b, 1984], Sacks *et al*[1989a, 1989b] and Currin *et al*[1991] estimates for the response functions given their models and "observations" $y_1, \ldots, y_n$ at $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$.

Sacks *et al*[1989a, 1989b] obtain the best linear estimate (which minimizes the mean square error)

$$\hat{Y}(\boldsymbol{x}) = \boldsymbol{f}^{\mathrm{T}}(\boldsymbol{x})\hat{\boldsymbol{\beta}} + \boldsymbol{v}^{\mathrm{T}}(\boldsymbol{x})V^{-1}(\boldsymbol{y} - F\hat{\boldsymbol{\beta}}) \tag{2.9}$$

where $\hat{\boldsymbol{\beta}} = (F^{\mathrm{T}}V^{-1}F)^{-1}F^{\mathrm{T}}V^{-1}\boldsymbol{y}$, with associated MSE,

$$MSE(\hat{Y}(\boldsymbol{x})) = \sigma^2 - \begin{pmatrix} \boldsymbol{f}^{\mathrm{T}}(\boldsymbol{x}) & \boldsymbol{v}^{\mathrm{T}}(\boldsymbol{x}) \end{pmatrix} \begin{pmatrix} 0 & F^{\mathrm{T}} \\ F & V \end{pmatrix}^{-1} \begin{pmatrix} \boldsymbol{f}(\boldsymbol{x}) \\ \boldsymbol{v}(\boldsymbol{x}) \end{pmatrix} \tag{2.10}$$

where

$$\begin{aligned} \boldsymbol{f}(\boldsymbol{x}) &= (f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))^{\mathrm{T}} \\ \boldsymbol{v}(\boldsymbol{x}) &= (\mathrm{Cov}(Y(\boldsymbol{x}), Y(\boldsymbol{x}_1)), \ldots, \mathrm{Cov}(Y(\boldsymbol{x}), Y(\boldsymbol{x}_n)))^{\mathrm{T}} \\ \boldsymbol{y} &= (y_1, \ldots, y_n)^{\mathrm{T}} \\ V_{ij} &= \mathrm{Cov}(Y(\boldsymbol{x}_i), Y(\boldsymbol{x}_j)) \\ F_{ij} &= f_j(\boldsymbol{x}_i) \end{aligned}$$

O'Hagan's comment on Sacks *et al*[1989b] includes a Bayesian justification, putting a multi-variate normal prior distribution on the regression parameters,

$$\boldsymbol{\beta} \sim \mathbb{N}(\boldsymbol{\beta}_0, B)$$

which leads to the same form of estimate, but with $\hat{\boldsymbol{\beta}}$ replaced by

$$\tilde{\boldsymbol{\beta}} = (F^{\mathrm{T}}V^{-1}F + \sigma^2 B^{-1})^{-1}(F^{\mathrm{T}}V^{-1}F\hat{\boldsymbol{\beta}} + \sigma^2 B^{-1}\boldsymbol{\beta}_0) \tag{2.11}$$

which is a weighted average of the prior mean, and the least-squares estimate. If we allow the prior variance matrix of the $\boldsymbol{\beta}$ to tend towards infinity, $\tilde{\boldsymbol{\beta}}$ tends to $\hat{\boldsymbol{\beta}}$.

In Schagen's model, the best estimate is

$$\hat{Y}(\boldsymbol{x}) = \boldsymbol{v}^{\mathrm{T}}(\boldsymbol{x})V^{-1}(\boldsymbol{y} - \mu\mathbf{1}) + \mu \tag{2.12}$$

with associated variance

$$\mathrm{Var}(\hat{Y}(\boldsymbol{x})) = \sigma^2(1 - \boldsymbol{v}^{\mathrm{T}}(\boldsymbol{x})V^{-1}\boldsymbol{v}(\boldsymbol{x})) \tag{2.13}$$

The maximum likelihood estimate for μ can be calculated and is given by

$$\hat{\mu} = \frac{\mathbf{1}^{\mathrm{T}}V^{-1}\boldsymbol{y}}{\mathbf{1}^{\mathrm{T}}V^{-1}\mathbf{1}} \tag{2.14}$$

and can be substituted back in the previous equation.

It would also be useful to find estimates for the other parameters θ and σ^2. In the model of Sacks *et al*[1989a, 1989b] the MLE of σ^2 is

$$\hat{\sigma}^2 = \frac{1}{n}(\boldsymbol{y} - F\hat{\boldsymbol{\beta}})^{\mathrm{T}}V^{-1}(\boldsymbol{y} - F\hat{\boldsymbol{\beta}}) \tag{2.15}$$

and they use cross validated mean square error to estimate θ. Currin *et al* use the maximum likelihood estimate for θ as well.

Schagen uses a cumulative predictor method instead of cross validated mean square error or maximum likelihood to estimate θ. He defines

$$\varepsilon_i = Y(\boldsymbol{x}_i) - \hat{Y}_{i-1}(\boldsymbol{x}_i)$$

where $\hat{Y}_{i-1}(\boldsymbol{x}_i)$ is the predicted value of $Y(\boldsymbol{x}_i)$ based on the first $i-1$ function evaluations. He shows the log likelihood of V is then proportional to

$$H = -\sum_{i=1}^{n} \frac{\varepsilon_i^2}{\sigma_i^2},$$

where the variance of these differences σ_i^2 can be found by generating the a new matrix

V^* from V by pivoting on the diagonal elements of V, as follows

$$\begin{aligned} V^*_{kk} &= -V^{-1}_{kk} \\ V^*_{kl} = V^*_{lk} &= -V^{-1}_{kk} V_{kl} \\ V^*_{jl} = V^*_{lj} &= V_{jl} - V_{jk} V^{-1}_{kk} V_{kl} \end{aligned}$$

He uses this new **matrix** to compute the ε_is and their variances

$$\begin{aligned} \varepsilon_i &= (Y(\boldsymbol{x}_j) - \mu) + \sum_{j=1}^{i-1} V^*_{ij}(Y(\boldsymbol{x}_j) - \mu) \\ \sigma^2_i &= V^*_{ii} \end{aligned}$$

and chooses θ to maximize H.

Schagen also examines another simple way of estimating θ, by comparing nearest neighbours. Given any design point $\boldsymbol{x}$, he finds the nearest design point to it $\boldsymbol{x}^*$, and works out the distance d between the two, (including in this any anisotropy parameters)

$$d^2 = \sum_{i=1}^{q} \alpha_i (x_i - x^*_i)^2.$$

In this simple case he works out the log-likelihood of the covariance matrix, $V = \sigma^2 \begin{pmatrix} 1 & e^{-\theta d^2} \\ e^{-\theta d^2} & 1 \end{pmatrix}$,

$$\log(l(V|\theta)) \propto -\frac{1}{2} log|V| - \frac{1}{2} \boldsymbol{y}^{\mathrm{T}} V^{-1} \boldsymbol{y}$$

where $\boldsymbol{y} = (y, y^*)^{\mathrm{T}}$. Maximizing this with respect to θ gives us the following cubic

$$w^3 - \frac{yy^*}{\sigma^2} w^2 + (\frac{y^2 + y^{*2}}{\sigma^2} - 1)w - \frac{yy^*}{\sigma^2} = 0$$

where $w = e^{-\theta d}$, which if it has a solution in $[0, 1]$ gives an estimate of θ, $-\frac{\log w}{d^2}$. He now has a collection of n estimates of θ for different design points, of which he uses the median as an estimate of θ.

Schagen extends his model by combining two stationary stochastic processes, where one represents a long range trend and the other more local departures from it, i.e.

$$Y(\boldsymbol{x}) = Z_L(\boldsymbol{x}) + Z_S(\boldsymbol{x}) + \mu$$

where $Z_L(\cdot)$ and $Z_S(\cdot)$ are both as above, but with $\theta_L \ll \theta_S$. The θ parameters of each process are estimated, for $Z_S(\cdot)$ by only considering design points that are very close together, and $Z_L(\cdot)$ by using the standard approaches above on the function $Y(\cdot) - Z_L(\cdot)$.

Schagen goes on to use his model for optimization, which we will look at in a later section, but does not discuss design issues in relation to the problem of interpolation. On the other hand Sacks *et al*[1989a, 1989b] do approach the problem of design. As they are working with expensive function, they would like to choose the design (the set $\mathcal{X}$ of $\boldsymbol{x}$s taken from a design region $\mathcal{E}$, at which to evaluate the function) which minimizes the number of runs that are required. We cannot know this exactly (as it would entail us knowing precisely where the zero is), so instead we need to look at other similar criteria. We are not interested here in the pros and cons of different design schemes, whether we choose all the points initially or choose the points sequentially, as we will be looking into this in more detail in Chapter 5 in relation to our problem. Suggestions for criteria to be minimized in their papers are:

EISE: Empirical Integrated Squared Error is used to compare different interpolants when developing models, to test how well an approximant fits the true function and is defined by

$$EISE[\hat{Y}] = \sum_i (\hat{Y}(w_i) - Y(w_i))^2$$

where the w_i are N randomly chosen points in the design space $\mathcal{E}$.

MESE: Maximum Empirical Squared Error, again used similarly to above, to compare different interpolants when developing models

$$MESE[\hat{Y}] = \max_i (\hat{Y}(w_i) - Y(w_i))^2.$$

Both this and EISE have no practical use with real problems but are useful in testing the methods, as they need extra function evaluations to calculate them.

IMSE: Integrated Mean Square Error over a region $\mathcal{E}$ is calculated by integrating the MSE of $\hat{Y}(\boldsymbol{x})$ obtained earlier, i.e. (For ease of calculation we note that $\boldsymbol{a}^{\mathrm{T}}B\boldsymbol{c} = \mathrm{trace}B\boldsymbol{c}\boldsymbol{a}^{\mathrm{T}}$)

$$IMSE[\hat{Y}] = \int_{\Omega} \sigma^2 d\boldsymbol{x} - \mathrm{trace}\left(\begin{pmatrix} 0 & F^{\mathrm{T}} \\ F & V \end{pmatrix}^{-1} \int_{\mathcal{E}} \begin{pmatrix} \boldsymbol{f}(\boldsymbol{x})\boldsymbol{f}^{\mathrm{T}}(\boldsymbol{x}) & \boldsymbol{f}(\boldsymbol{x})\boldsymbol{v}^{\mathrm{T}}(\boldsymbol{x}) \\ \boldsymbol{v}(\boldsymbol{x})\boldsymbol{f}^{\mathrm{T}}(\boldsymbol{x}) & \boldsymbol{v}(\boldsymbol{x})\boldsymbol{v}^{\mathrm{T}}(\boldsymbol{x}) \end{pmatrix} d\boldsymbol{x} \right)$$

the multiple integrals in the trace simplify if we assume that all the functions $f_i(\cdot)$ in the regression terms, and all the correlation functions are factorizable into terms containing only one co-ordinate, and that they are independent of the design so only need calculating once.

MMSE: Maximum Mean Square Error. Instead of finding the average value of the MSE, we find its maximum value, although we no longer have any integration to perform we do have to find the global maxima of the MSE for every design we choose – minimax conditions are always hard to find, as they require two optimization stages.

Entropy: Posterior Entropy is an idea put forward by Lindley[1956] in work on Bayesian design. It quantifies the "amount of information" in an experiment. If observations are taken at a set of design points $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ in $\mathcal{E}$, then the posterior entropy is defined as $\mathrm{E}(-\log p)$, where p is the conditional density of $Y(\cdot)$ on $X' = \mathcal{E} - X$ given observations $y_1, \ldots, y_n$. It can be shown that minimizing the expected posterior entropy on X' is the same as maximizinig the prior variance on X. If we assume normality this is the same as maximising the determinant of the covariance matrix of responses. In the Schagen model covered by Currin *et al*[1991], this is the same as maximizing the determinant of V. This criteria has a tendency to place points as far as possible away from each other as possible, and this is similar to the criteria to be discussed in Chapter 4.

Although not considered in any of the papers cited above the averaging criteria

EISE, IMSE and entropy, can be averaged with respect to a known non-uniform weight function, which will produce different designs, see Chapter 4.

Sacks *et al* perform a robustness study choosing designs which are near optimal for a range of values of θ.

2.3 Integration

In this section we consider two more approaches to Bayesian integration – we have already encountered one in Section 2.2.2. Firstly we can take the interpolant, as defined in Section 2.2.3, and integrate it as an approximation to the integral, this is the approach mentioned by Schagen[1980a, 1980b], and O'Hagan[1990]. Eberlain on the other hand uses an approach similar to Poincaré, by placing a prior on the coefficients of the function's power series.

2.3.1 Bayesian quadrature

Although Schagen only mentions integration in passing, O'Hagan looks at integration in more detail. He considers integrals of the form

$$\int_{\mathcal{X}} r(\boldsymbol{x})Y(\boldsymbol{x})d\Omega_{\mathcal{X}}(\boldsymbol{x})$$

where $\Omega_{\mathcal{X}}(\boldsymbol{x})$ is some measure over the design space $\mathcal{X}$, and $r(\boldsymbol{x})$ is a product of polynomials in the x_is, i.e. $r(\boldsymbol{x}) = r^{(1)}(x_1)\cdots r^{(n)}(x_n)$. The functions he is considering are unnormalized posterior probability densities, and so is interested in their integral – the constant of proportionality in the Bayes analysis; the ratios of two integrals of the form

$$\frac{\int_{\mathcal{X}} r(\boldsymbol{x})Y(\boldsymbol{x})d\Omega_{\mathcal{X}}(\boldsymbol{x})}{\int_{\mathcal{X}} Y(\boldsymbol{x})d\Omega_{\mathcal{X}}(\boldsymbol{x})}$$

– which give "moments" of the distribution $Y(\cdot)$ along with other summary statistics, which are combinations of these integrals.

O'Hagan models the function in a similar manner to Sacks *et al* but from a

Bayesian viewpoint. So he assumes that $Y(x)$ has the following prior

$$Y(\boldsymbol{x})|\boldsymbol{\beta},\sigma^2 \sim \mathbb{N}(\boldsymbol{f}^{\mathrm{T}}(\boldsymbol{x})\boldsymbol{\beta},\sigma^2\rho_2(\boldsymbol{x},\boldsymbol{x}))$$

and places an improper "ignorance" prior on the parameters

$$\pi(\boldsymbol{\beta},\sigma^2) \propto \sigma^{-2}$$

and he defines

$$\boldsymbol{k} = \int_{\mathcal{X}} \boldsymbol{r}(\boldsymbol{x})Y(\boldsymbol{x})d\Omega_{\mathcal{X}}(\boldsymbol{x}) \tag{2.16}$$

where $\boldsymbol{r}(\boldsymbol{x}) = (r_1(\boldsymbol{x}),\ldots,r_p(\boldsymbol{x}))^{\mathrm{T}}$. Using the notation from Section 2.2.3, he finds posterior distributions for the parameters

$$\boldsymbol{\beta}|\boldsymbol{y},\sigma^2 \sim \mathbb{N}(\hat{\boldsymbol{\beta}},(F^{\mathrm{T}}V^{-1}F)^{-1}) \tag{2.17}$$

$$\sigma^2|\boldsymbol{y} \sim d\chi^{-2}_{n-p} \tag{2.18}$$

(where d is calculated from the data) and for $Y(x)$. He then integrates his interpolant, giving us a posterior distribution for $\boldsymbol{k}$

$$\boldsymbol{k}|\boldsymbol{y} \sim t_{n-m}(\hat{\boldsymbol{k}},dW) \tag{2.19}$$

where $\hat{\boldsymbol{k}}$ and W can be calculated from the data also, and t_{n-p} is the multivariate t-distribution with $n-p$ degrees of freedom.

In general when considering the distribution of the ratio of two random quantities, we usually find it is not well defined because of the possibility that the denominator is zero. However, in the case where O'Hagan is considering, the denominator should never be zero as it is usually the integral of a strictly positive function, (sometimes this integration can be zero or negative as the approximant can be negative) . If we have made sufficient observations then the variation of the denominator should be sufficiently small to obtain a reasonably accurate approximation to the ratio, using

the "delta-methods", the posterior mean and variance of $\frac{k_i}{k_j}$ can be approximated by

$$\mathrm{E}\left(\frac{k_i}{k_j}|\boldsymbol{y}\right) \simeq \frac{\mathrm{E}(k_i|\boldsymbol{y})}{\mathrm{E}(k_j|\boldsymbol{y})}\left(1+\frac{\mathrm{Var}(k_j|\boldsymbol{y})}{\mathrm{E}(k_j|\boldsymbol{y})^2}-\frac{\mathrm{Cov}(k_i,k_j|\boldsymbol{y})}{\mathrm{E}(k_i|\boldsymbol{y})\mathrm{E}(k_j|\boldsymbol{y})}\right) \tag{2.20}$$

$$\mathrm{Var}\left(\frac{k_i}{k_j}|\boldsymbol{y}\right) \simeq \left(\frac{\mathrm{E}(k_i|\boldsymbol{y})}{\mathrm{E}(k_j|\boldsymbol{y})}\right)^2\left(\frac{\mathrm{Var}(k_j|\boldsymbol{y})}{\mathrm{E}(k_j|\boldsymbol{y})^2}+\frac{\mathrm{Var}(k_i|\boldsymbol{y})}{\mathrm{E}(k_j|\boldsymbol{y})^2}-2\frac{\mathrm{Cov}(k_i,k_j|\boldsymbol{y})}{\mathrm{E}(k_i|\boldsymbol{y})\mathrm{E}(k_j|\boldsymbol{y})}\right) \tag{2.21}$$

To complete this O'Hagan now needs to specify the constituents of the model: the measure $\Omega_{\mathcal{X}}(\cdot)$ over which to integrate; the correlation function $\rho(\cdot)$; the linear regression terms $f_i(\cdot)$; and the functions he wishes to find posterior expectations of $r_i(\cdot)$. He needs to consider functions which allow him to compute his estimates for $\hat{\boldsymbol{k}}$ in closed form, otherwise he would still have to resort to numerical integration. He assumes that he is integrating over $\mathcal{X} = \mathbb{R}^q$; that $\Omega_{\mathcal{X}}(\boldsymbol{x})$ is the standard normal distribution $\mathbb{N}(0, I_q)$; and that the correlation function is $\rho_2(\cdot)$. As well he assumes that all the $f_i(\cdot)$s and $r_i(\cdot)$s are simple products of polynomials, this implies that he can obtain closed form expressions for each of the integrals.

O'Hagan suggests two design criteria for choosing the initial design $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ both of which minimize the variance of one of the quantities of interest we mentioned above

$\mathcal{C}_1$ minimize the variance of one of the ratios of integrals – for example the posterior mean of the distribution $Y(x)$

$$\frac{\int_{\mathcal{X}} r(\boldsymbol{x})Y(\boldsymbol{x})d\Omega_{\mathcal{X}}(\boldsymbol{x})}{\int_{\mathcal{X}} Y(\boldsymbol{x})d\Omega_{\mathcal{X}}(\boldsymbol{x})}$$

$\mathcal{C}_2$ minimize the variance of the denominator in one of the ratios, often

$$\int_{\mathcal{X}} Y(\boldsymbol{x})d\Omega_{\mathcal{X}}(\boldsymbol{x})$$

the constant of proportionality in the Bayes analysis.

Marginal and conditional distributions derived from the posterior distribution $Y(\cdot)$ are also of interest in Bayesian analysis and these can be obtained in a similar way.

O'Hagan factorises $\mathcal{X}$ into $\mathcal{X}_a$ and $\mathcal{X}_b$ (putting $\boldsymbol{x}^{\mathrm{T}} = (\ \boldsymbol{x}_a^{\mathrm{T}}\ \ \boldsymbol{x}_b^{\mathrm{T}}\ )^{\mathrm{T}}$) and, for simplicity, chooses a measure that will factorise, i.e. $d\Omega_{\mathcal{X}}(\boldsymbol{x}) = d\Omega_{\mathcal{X}_a}(\boldsymbol{x}_a)d\Omega_{\mathcal{X}_b}(\boldsymbol{x}_b)$, so he can write

$$k \stackrel{\text{def}}{=} \int_{\mathcal{X}} Y(\boldsymbol{x})d\Omega_{\mathcal{X}}(\boldsymbol{x}) = \int_{\mathcal{X}_a}\left(\int_{\mathcal{X}_b} Y(\boldsymbol{x}_a,\boldsymbol{x}_b)d\Omega_{\mathcal{X}_b}(\boldsymbol{x}_b)\right)d\Omega_{\mathcal{X}_a}(\boldsymbol{x}_a)$$

He also defines

$$p_{\mathcal{X}_a}(\boldsymbol{x}_a) \stackrel{\text{def}}{=} \int_{\mathcal{X}_b} Y(\boldsymbol{x}_a,\boldsymbol{x}_b)d\Omega_{\mathcal{X}_b}(\boldsymbol{x}_b)$$

which is the constant of proportionality for the conditional distribution, given he has observed $\boldsymbol{x}_a$. He would like to make inferences about $p_{\mathcal{X}_a}(\boldsymbol{x}_a)$, the marginal density $p_{\mathcal{X}_a}(\boldsymbol{x}_a)/k$ and the conditional densities $Y(\boldsymbol{x})/p_{\mathcal{X}_a}(\boldsymbol{x}_a)$, the general integrals

$$\boldsymbol{k}_{\mathcal{X}_a}(\boldsymbol{x}_a) \stackrel{\text{def}}{=} \int_{\mathcal{X}_b} \boldsymbol{r}(\boldsymbol{x}_a,\boldsymbol{x}_b)Y(\boldsymbol{x}_a,\boldsymbol{x}_b)d\Omega_{\mathcal{X}_b}(\boldsymbol{x}_b)$$

or ratios of the form $k_{\mathcal{X}_a,i}(\boldsymbol{x}_a)/p_{\mathcal{X}_a}(\boldsymbol{x}_a)$ or $k_{\mathcal{X}_a,i}(\boldsymbol{x}_a)/k$.

He then repeats the calculations made above to obtain posterior distributions for this vector.

$$\boldsymbol{k}_{\mathcal{X}_a}(\boldsymbol{x}_a)|\boldsymbol{y},\sigma^2 \sim \mathbb{N}(\hat{\boldsymbol{k}}_{\mathcal{X}_a}(\boldsymbol{x}_a),\sigma^2 W_{\mathcal{X}_a}(\boldsymbol{x}_a,\boldsymbol{x}_a)) \tag{2.22}$$

$$\boldsymbol{k}_{\mathcal{X}_a}(\boldsymbol{x}_a)|\boldsymbol{y} \sim t_{n-q}(\hat{\boldsymbol{k}}_{\mathcal{X}_a}(\boldsymbol{x}_a), dW_{\mathcal{X}_a}(\boldsymbol{x}_a,\boldsymbol{x}_a)) \tag{2.23}$$

where again $\hat{\boldsymbol{k}}_{\mathcal{X}_a}(\boldsymbol{x}_a)$ and $W_{\mathcal{X}_a}(\boldsymbol{x}_a,\boldsymbol{x}_a)$ are functions of the data.

From this he obtains posterior estimates of the conditional moments as ratios $\frac{k_{\mathcal{X}_a,i}(\cdot)}{p_{\mathcal{X}_a}(\cdot)}$, using the "delta-method" approximation as above. To get the marginal distributions he needs to approximate $\frac{p_{\mathcal{X}_a}(\cdot)}{k}$, the only additional information needed is the covariance between $\boldsymbol{k}_{\mathcal{X}_a}(\boldsymbol{x}_a)$ and $\boldsymbol{k}$, which can again be calculated as above.

O'Hagan concludes the paper by considering product designs, as the matrix algebra can be simplified if we assume

(1) the design we are considering consists of a lattice of points, i.e. $\{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n\}$ consists of all $n_a n_b$ points of the form $(\boldsymbol{x}_{a_i}^{\mathrm{T}},\boldsymbol{x}_{b_j}^{\mathrm{T}})$

(2) the correlation function $\rho(\cdot)$ can be factorised, i.e.

$$\rho(\boldsymbol{x} - \boldsymbol{x}_*) = \rho_{\mathcal{X}_a}(\boldsymbol{x}_a - \boldsymbol{x}_{*a})\rho_{\mathcal{X}_b}(\boldsymbol{x}_b - \boldsymbol{x}_{*b})$$

(3) $\boldsymbol{f}$ consists of all $m_a m_b$ products of m_a functions of $\boldsymbol{x}_a$ and m_b functions of $\boldsymbol{x}_b$

(4) Similarly for $\boldsymbol{r}$ (if this condition does not hold, we can always extend the set of functions we are integrating to satisfy it)

All the matrices in the calculations can then be factorized into Krönecker products, so simplifying the equations.

2.3.2 Stick breaking

Another Bayesian approach to numerical integration is that of Eberlain. He considers functions on the interval $[-1, 1]$, which have convergent power series, in a similar way to Poincaré, identifying each function $f(\cdot)$ with its set of coefficients.

$$f(x) = \sum_{n=0}^{\infty} C_n x^n$$

He then place a prior on these coefficients:

$$\begin{aligned}
C_0 &\sim U(-1, 1) \\
C_1|C_0 &\sim U(-[1 - |C_0|], [1 - |C_0|]) \\
&\vdots \quad \vdots \\
C_{n+1}|C_n, \ldots, C_0 &\sim U(-[1 - |C_0| - \cdots - |C_n|], [1 - |C_0| - \cdots - |C_n|]) \\
&\vdots \quad \vdots
\end{aligned}$$

This sets up a prior on the functions bounded above and below (by 1 and -1) such that $\sum_{n=0}^{\infty} |C_n| = 1$. He extends this to functions bounded above and below by arbitrary values, by shifting and re-scaling. This prior is then used to approximate the integral by obtaining posterior expectations $\{c_0, c_1, \ldots\}$ for the coefficients $\{C_0, C_1, \ldots\}$, and

then computing the integral, of the resulting power series.

$$\begin{aligned} \mathrm{E}\left(\int_{-1}^{1} Y(x)dx\right) &= \int_{-1}^{1} \sum_{n=0}^{\infty} c_n x^n \\ &= 2\sum_{m=0}^{\infty} \frac{c_{2m}}{2m+1} \end{aligned}$$

2.4 Optimization

An obvious approach to finding local or global optima, is to locate the local or global optima for the approximant, as Schagen[1984] and Möckus[1989] do.

Schagen[1980b, 1984], Sacks *at al*[1989a, 1989b] and Currin *at al*[1991] maximize their interpolant to obtain an estimate of the global maximum of the real function. However, Schagen[1984] includes a sequential design criterion for choosing the next point to evaluate the function at. The criterion is a composite function which is a compromise between looking near the current maxima and looking in any gaps in the design to see if a maxima has been missed. He considers functions defined in a rectangular region $(a_1, b_1) \times \ldots \times (a_q, b_q)$, with the criterion to maximize,

$$G(\boldsymbol{x}^*) = W\frac{\hat{Y}(\boldsymbol{x}^*)}{\sigma} - (1-W)\frac{H(\boldsymbol{x}^*)}{H_m} \tag{2.24}$$

using the same notation as in Section 2.2, and where

$$\begin{aligned} H(\boldsymbol{x}^*) &= \sum_{i=1}^{n} \rho(\boldsymbol{x}^* - \boldsymbol{x}_i) + \sum_{k=1}^{p} e^{-4\theta(a_k - x_k^*)^2} + e^{-4\theta(b_k - x_k^*)^2} \\ H_m &= e^{-\frac{\theta}{2}\left(\frac{\prod_{k=1}^{p} b_k - a_k}{n}\right)^{\frac{1}{p}}} \end{aligned}$$

For small W (near 0), the second term is dominant, and this "repulsive" function chooses design points that are as far away from the other design points (the first summation in $H(\boldsymbol{x}^*)$) and the edge of the region as possible (the second summation in $H(\boldsymbol{x}^*)$). Therefore the criterion "explores" the region. For W near one, the first

term becomes dominant, and so the next design points will be chosen to be close to the maxima of the interpolant. Letting W increase from 0 to 1 he generates a design, which initially explores the region and then homes in on the maximum. A fixed scheme could be used for the values of W, but it would be better to choose it when more is known about the function. One scheme is to let W be the "probability" that there are no more undetected "lumps".

Möckus'[1989] approaches the problem of global optimization (minimization) of functions defined on the rectangular region $[-1,1]^q$ in a similar manner. He defines the "distance" of a point in $\boldsymbol{x} \in \mathcal{X} = [-1,1]^q$ from the global minima $\boldsymbol{x}_0$ to be $y(\boldsymbol{x}) - y(\boldsymbol{x}_0)$, rather than the euclidean distance $|\boldsymbol{x} - \boldsymbol{x}_0|$. This function is well defined even if $y(\cdot)$ has multiple global minima.

He considers two design criteria. The first a "worst-case" criterion is defined to be the maximum "distance" from the global minimum over functions $y(\cdot)$ in the possible space of functions $\mathcal{Y}$. The second one, which he uses in the book, is the average "distance" from the global minima, where the average is taken over functions in $\mathcal{Y}$, giving the criterion

$$\mathcal{C}(\boldsymbol{x}) = \int_{\mathcal{Y}} |y(\boldsymbol{x}) - y(\boldsymbol{x}_0)| d\Omega(y)$$

where $\Omega(\cdot)$ is some additive measure over $\mathcal{Y}$.

His choice of prior distribution for the functions is influenced by conditions of continuity, and independence of partial derivatives. These arguments lead him to choosing the prior distribution to be a stationary gaussian process with mean μ and covariance

$$\mathrm{Cov}(Y(\boldsymbol{x}, Y(\boldsymbol{x}_*)) = \sigma^2 \prod_{i=1}^{q} \left(1 - \frac{|x_i - x_{*i}|}{2}\right)$$

Given data $y_1, \ldots, y_n$ at $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, he obtains the posterior distribution of the functions in $\mathcal{Y}$, uses this as $\Omega(\cdot)$ to compute $\mathcal{C}(\boldsymbol{x})$, and chooses the next point to minimize this criterion.

2.5 Smoothing

If the function evaluations are subject to error (not necessarily random), we require a different approach because the approximant no longer matches the true function at the design points.

If we assume this error is truly random we can expand the work on interpolation by stationary stochastic processes by adding a further independent error term to the $Z(\cdot)$ process with variance σ_ε^2, this gives us a modified covariance structure

$$\mathrm{Cov}(Z(\boldsymbol{x}_i), Z(\boldsymbol{x}_j)) = \sigma^2\rho(\boldsymbol{x}_i - \boldsymbol{x}_j) + \sigma_\varepsilon^2\delta_{ij}$$

Here we call the approximants "smoothers" instead of interpolants. We do not necessarily need to assume the errors independent, but can use a very short range correlated error, as in Schagen's model, and use the long range term $Z_L(\cdot)$ as the approximant.

The methods of Section 2.2.2 can be extended to smoothing, see Wahba[1978]. She and other authors use a penalty function, of the form

$$L[y] = \alpha \int y''(x)^2 dx + \frac{1}{n}\sum (y_i - y(x_i))^2$$

with observations $y_1, \ldots, y_n$ at $x_1, \ldots, x_n$. This penalty function is a trade off between fidelity to the data and smoothness of the approximation, α measures the smoothness of the response (large α means the function is very smooth, small α means the function has high fidelity to the data). It is easy to show that the function that minimizes $L[\cdot]$ is a cubic spline.

Finally in this section, we look at O'Hagan's[1978] approach to smoothing using a local regression model, in which he generalizes the standard linear regression model to allow the regression parameters to vary with x. In a standard linear regression we model the response $Y(x)$ at x as follows:

$$\mathrm{E}(Y(x)|x, \boldsymbol{\beta}) = \boldsymbol{f}(x)^{\mathrm{T}}\boldsymbol{\beta} \tag{2.25}$$

where $\boldsymbol{f}(x)$ is a vector of known functions and $\boldsymbol{\beta}$ is a vector of unknown parameters. O'Hagan generalizes equation (2.25) by allowing the regression parameters to depend on x as well

$$\mathrm{E}(Y(x)|x, \boldsymbol{\beta}(x)) = \boldsymbol{f}(x)^{\mathrm{T}}\boldsymbol{\beta}(x), \tag{2.26}$$

and assumes the errors are homoscedastic

$$\mathrm{Var}(Y(x)|x, \boldsymbol{\beta}(x)) = \sigma^2 \tag{2.27}$$

O'Hagan lays down a prior distribution on the regression parameters, by assuming that

$$\mathrm{E}(\boldsymbol{\beta}(x)|\boldsymbol{b}_0) = \boldsymbol{b}_0 \tag{2.28}$$

and that it is a second-order stationary process,

$$\mathrm{Cov}(\boldsymbol{\beta}(x), \boldsymbol{\beta}(x^*)|\boldsymbol{b}_0) = \rho_2(|x - x^*|)B_0 \tag{2.29}$$

where $\rho_2(\cdot)$ is as in the previous section, and $\boldsymbol{b}_0$ and B_0 are the mean and variance of $\boldsymbol{\beta}(x)$. He completes the prior distribution by assuming the $\boldsymbol{\beta}(x)$s are multivariate normal.

He then obtains posterior distributions for the parameters, given he has observation $y_1, \ldots, y_n$ at $x_1, \ldots, x_n$

$$\boldsymbol{\beta}(x) \sim \mathbb{N}(\boldsymbol{b}_1(x), B_1(x, x)) \tag{2.30}$$

where $\boldsymbol{b}_1(x)$ and $B_1(x, x)$ are the posterior mean and variance of $\boldsymbol{\beta}(x)$, and can be computed from the data, this is then used to obtain the posterior distribution for the response

$$Y(x) \sim \mathbb{N}(\boldsymbol{f}(x)^{\mathrm{T}}\boldsymbol{b}_1(x), \sigma^2 + \boldsymbol{f}(x)^{\mathrm{T}}B_1(x, x)\boldsymbol{f}(x)) \tag{2.31}$$

This approximant becomes an interpolant when the error variance σ^2 is equal to zero.

He uses the mean and variance of this predictor as an approximation $y(\cdot)$ and a

measure of error respectively. If there is little or no information available about the regression parameters, it is possible to "plead ignorance" about them by placing an improper prior distribution on the prior mean vector $\boldsymbol{b}_0$, i.e let $\boldsymbol{b}_0$ be distributed as

$$\boldsymbol{b}_0 \sim \mathbb{N}(\boldsymbol{b}^*, kB^*) \tag{2.32}$$

and let $k \to \infty$, he then obtains the following posterior distribution for the parameters

$$\boldsymbol{\beta}(x) \sim \mathbb{N}(\boldsymbol{b}_2(x), B_2(x,x)) \tag{2.33}$$

where again $\boldsymbol{b}_2(x)$ and $B_2(x,x)$ can be calculated from the data. The posterior distribution of $Y(x)$ can again be found

$$Y(x) \sim N(\boldsymbol{f}(x)^{\mathrm{T}}\boldsymbol{b}_2(x), \sigma^2 + \boldsymbol{f}(x)^{\mathrm{T}}B_2(x,x)\boldsymbol{f}(x)) \tag{2.34}$$

$\boldsymbol{f}(x)^{\mathrm{T}}\boldsymbol{b}_2(x)$ can be used as a "smoother".

O'Hagan digresses by using these models as a basis to find a simple linear predictor of the form $\hat{y}(x) = \boldsymbol{h}(x)^{\mathrm{T}}\boldsymbol{\gamma}$, by choosing the parameters $\boldsymbol{\gamma}$ to minimize

$$L(\boldsymbol{\gamma}) = \int d\Omega(x)\mathrm{E}((y(x) - \hat{y}(x))^2)$$

where $\Omega(x)$ is a measure expressing the probability that a future prediction will need to be made with a given x-value. As $y(x)$ is not known exactly $\boldsymbol{\gamma}$ is found to minimize the loss over all possible functions using either of the posterior distributions obtained above for $y(\cdot)$. He chooses designs which minimize the expected loss of this criteria, before he makes any function evaluations.

2.6 Model inadequacy

Blight and Ott[1975] use the stationary stochastic process approach to analyse systematic departures from a linear regression model, by using a model similar to that

in Sacks *et al*[1989a, 1989b], including a random error term.

$$Y(x_i) = \boldsymbol{f}(x_i)\boldsymbol{\beta} + Z(x_i) + \varepsilon(x_i)$$

They obtain estimates for the $\boldsymbol{\beta}$ parameters, and then use the additional terms from the $Z(\cdot)$ process to measure the inadequacies of the fit. They use this model to choose the prediction weights for linear estimates of the function at various points. Their weights put more emphasis on observations near the point where the function is to be approximated, whereas in the classical least squares prediction the weighting is spread more evenly over the whole of the design region.

Chapter 3

Bayes Linear and Partial Belief Specification

Probability doesn't exist — B. de Finetti. (Theory of Probability[1970])

In this chapter I will endeavour to explain the subjectivist methodology of limited belief specification. In the subjective methodology, instead of probabilities being derived from limiting frequency style arguments, the person wishing to analyse his beliefs, whom I shall term *You*, as in de Finetti[1970], lays down his own set of probabilities (in the 'classic' approach) or his own set of previsions (expectations) of the quantities of interest.

In a 'classic' full Bayesian analysis we are required to specify full prior distributions for any quantity we want to make inference about, and to specify the full conditional distributions needed for the likelihood of each observable, and then apply Bayes theorem,

$$\pi_*(x|y) = \frac{l(y|x)\pi_0(x)}{\int l(y|t)\pi_0(t)dt}$$

to get the posterior distributions.

This methodology has many disadvantages. From a computational point of view, the formulae for the posterior distributions are not, in general, expressible in closed

form, but instead numerical methods have to be used to compute these integrals. A more important, and often cited argument against Bayesian methods, is the practical and philosophical question of whether or not the prior distributions for the quantities and likelihood functions can be elicited from the beliefs held by *You*.

Consider the following problem; what is your full prior distribution for your height (H)? Could you for example specify what value you would put as your expectation of H^{107}, or specify the value you would assign for the probability that your height is in the interval $[h, h + \Delta_h)$ for all values of h? These are both very difficult (nearly impossible to do) but, these are just two of (a possibly infinite) set of judgements you are making when you produce the full prior distribution.

Often to get around this we make some approximation, for example, assume H is distributed normally with an appropriate mean and standard deviation. This is obviously only an approximation, as you are then allowing your height to possibly take negative values or large positive values. Usually the form of prior distribution and likelihood are chosen to make the algebra simple enough to make the posterior calculations tractable.

More importantly do you actually need to specify these quantities, are you ever really interested in what the 107^{th} moment of your height is, or what the probability that your height is between $6'$ and $6'\frac{1}{1024}''$ is. Therefore it is much more sensible to specify just those beliefs you actually hold and/or require for the problem, which leads on to the topic of limited or partial belief specification. The rest of this chapter summarises parts of this theory which will be required later in Chapters 4 and 5, and Appendix A contains a few additional results, and examples.

3.1 Prevision

For the collection of random quantities $\{X_1, X_2, \ldots\}$ of interest we can define the set $\mathcal{Q}$ of possible values that these quantities can take, we can think of each possible outcome as an ordered n-tuple, and $\mathcal{Q}$ can be thought of as being embedded in $\mathcal{A}$ the

linear ambit, a linear affine space. We can also consider the dual linear space $\mathcal{L}$, which consists of linear combinations of the quantities, i.e. $X = (a_1, a_2, \ldots) \in \mathcal{L}$ represents the linear combination $X = a_1X_1 + a_2X_2 + \cdots$, and in what follows I will refer to X in either of these forms, or even both simultaneously. It should be noted here that the representation of X as $(a_1, a_2, \ldots)$ is not necessarily unique as some of the X_is may be logically dependent, for example if $X_1 + X_2 = X_3$, then $(a_1, a_2, a_3, a_4, \ldots)$ can also be represented as $(a_1 + a_3, a_2 + a_3, 0, a_4, \ldots)$.

Now for each of the quantities X we can define its *prevision*, intuitively as the size of fixed gain *You* would consider equivalent to a random gain of size X, which we will denote by $\mathrm{P}(X)$. If we assume countable additivity, this has the same value as the expectation of X if you had laid down a full distribution for X.

We say a set of prevision statements is coherent if *You* have chosen them such that they do not show certain loss.

This quantity then satisfies the following properties:

(a) The prevision $\mathrm{P}(\cdot)$ is an additive function:

$$\mathrm{P}(X + Y) = \mathrm{P}(X) + \mathrm{P}(Y)$$

(If *You* are indifferent to the swap of X for $\mathrm{P}(X)$ and Y for $\mathrm{P}(Y)$ then *You* are also indifferent to the swap of $X + Y$ for $\mathrm{P}(X) + \mathrm{P}(Y)$.)

(b) The coherent prevision $\mathrm{P}(\cdot)$ satisfies :

$$\inf X \leq \mathrm{P}(X) \leq \sup X$$

(This is again obvious as if your choice is outside these values you would show certain loss.)

From these two properties it can be seen that $\mathrm{P}(\cdot)$ is a linear function i.e.

$$\mathrm{P}(aX) = a\mathrm{P}(X)$$

or more generally

$$\mathrm{P}(\sum_{i=1}^{n} a_i X_i) = \sum_{i=1}^{n} a_i \mathrm{P}(X_i).$$

so it can be seen that the set of random quantities for which we can obtain previsions for, given we have already declared our previsions of $X_1, X_2, \ldots$ is precisely those elements of $\mathcal{L}$ (which have a finite number of non-zero a_i's – this clarification is needed if we only assume finite additivity).

If we define $\mathcal{P}$ to be the set of coherent previsions, (those which we can choose without showing certain loss), then it can be readily proven to be the closed convex hull of $\mathcal{Q}$ in $\mathcal{A}$.

The above definition is not entirely useful, as although it is an obvious and intuitive definition, it is difficult to handle analytically, so we introduce some new alternative but equivalent definitions.

Defn. 3.1[a] *The* Prevision *of* X, $\mathrm{P}(X)$ *is defined to be the value of* $\overline{x}$ *which* You *would choose, so that after* You *have made this choice* You *are committed to accepting any bet whatsoever with gain* $c(X - \overline{x})$, *where* c *is arbitrary (positive or negative) and not at your control.*

Defn. 3.2[a] *Under definition 3.1[a], a set of your previsions is said to be* coherent *if among the set of bets* You *have committed yourself to accept, there are none for which the gains are all uniformly negative.*

Defn. 3.1[b] *The* Prevision *of* X, $\mathrm{P}(X)$ *is defined to be the value of* $\overline{x}$ *which in your opinion is the best choice if confronted with a penalty* L *proportional to the squared deviation of* X *from* $\overline{x}$

$$L = \left(\frac{X - \overline{x}}{k}\right)^2$$

(where k *is arbitrary, but previously fixed)*

Defn. 3.2^b *Under definition 3.1^b, a set of your previsions is said to be* coherent *if there is no other possible choice for your previsions that would lead to a uniform reduction in your penalty.*

It can be easily verified that these two pairs of definitions are equivalent to each other and equivalent to the intuitive definitions. Although of the pair the first is closer to the intuitive definition, the second is a more practical definition.

Now we can also introduce the notion of probability, the probability of an event H occurring can be considered to be the prevision of the indicator function I_H (or for simplicity H) which is 1 if H occurs and 0 if it does not. So even though we defined prevision without at first defining probability, we have effectively defined probability, and will use the same notation $\mathrm{P}(\cdot)$ for both.

3.2 Conditioning

We can define the *conditional prevision* of a quantity X given the event H to be the 'called-off' penalty equivalent of Definition 3.1^b. So we define,

Defn. 3.3 *The* conditional prevision $\mathrm{P}(X|H)$ *is the value of x we would choose if we were to incur the penalty $k^{-2}(X-x)^2$ if H occurs and 0 if it does not, i.e. the value of x we would choose if we were to incur the penalty $k^{-2}H(X-x)^2$.*

This notion of a 'called-off' bet can be used to define the set of conditional previsions $\{\mathrm{P}(X|H_1),\ldots,\mathrm{P}(X|H_n)\}$ over a partition $\mathcal{H}=\{H_1,\ldots,H_n\}$. If we define the penalty $L_{\mathcal{H}}$ as

$$L_{\mathcal{H}}=\frac{1}{k^2}(X-x_1H_1-\cdots-x_nH_n)^2$$

and observing that $H_1+\cdots+H_n=1$, and $H_iH_j=\delta_{ij}H_i$, the set of quantities $\{x_1,\ldots,x_n\}$ is precisely the set of conditional previsions.

This notion can be extended to any set of random quantities, as we no longer need to separate events from random quantities in general, we can make our choice

of $\{x_1, \ldots, x_n\}$ which would minimize the penalty L^* defined to be

$$L^* = \frac{1}{k^2}(X - x_1X_1 - \cdots - x_nX_n)^2$$

This development lies at the heart of Goldstein's notion of *Limited Belief Specification* [1981, 1988a, 1988b, 1991] which I will now summarise in the following sections.

3.3 Belief structures

The most logical way to combine collections of prevision statements is into inner product spaces, which are called *Belief Structures*

Defn. 3.4 *A* belief structure *A is defined as follows:*

a) Start with a (not necessarily finite) set of random quantities $C = \{X_0, X_1, \ldots\}$, *which includes* $X_0 = 1$, *the unit constant, and all* X_i *satisfy* $\mathrm{P}(X_i^2) < \infty$. *This is the* base *of A and written* $C = b(A)$.

b) Define $\mathcal{L}$ *as above, and define the inner product and norm over the equivalence classes of* $\mathcal{L}$ *(where* X *is related to* Y *if the prevision of* $(X - Y)^2$ *is zero) by*

$$(X, Y) = \mathrm{P}(XY), \qquad \|X\| = \sqrt{(X, X)}$$

Two belief structures (R and S) are the same ($R = S$) if they have a common base and inner product.

Belief structures can be added, to increase the level of detail of our specifications. If for example A and B are combined to produce $D = A + B$, we have to, in addition, specify $\mathrm{P}(XY)$ for each $X \in b(A)$ and $Y \in b(B)$. The base of D, $b(D)$ is simply $b(A) \cup b(B)$.

We will now introduce a simple example which will be continued throughout the rest of this chapter. To highlight the example it will be indented and set in *slanted* text.

We will consider a simple linear regression. Firstly we construct our model: we take n observations $Y_1, \ldots, Y_n$ *at design points* $x_1, \ldots, x_n$, *and we believe that*

$$Y_i = A + Bx_i + Z_i, \qquad \text{for } i = 1, \ldots, n, \tag{3.1}$$

where A, B *and the* Z_i*s are unknown quantities. We assume that the error terms (the* Z_i*s) are independent with mean zero and common variance* s^2. *We then lay down our beliefs about the parameters of the model.*

$$\begin{aligned} \mathrm{P}(A) &= a & \mathrm{P}(A^2) &= a^2 + s^2\sigma^2 \\ \mathrm{P}(B) &= b & \mathrm{P}(B^2) &= b^2 + s^2\tau^2 \\ & & \mathrm{P}(AB) &= ab + s^2\sigma\tau\rho \end{aligned}$$

We can then work out our implied beliefs about the observations

$$\begin{aligned} \mathrm{P}(Y_i) &= a + bx_i \\ \mathrm{P}(Y_iY_j) &= (a + bx_i)(a + bx_j) + s^2(\sigma^2 + \sigma\tau\rho(x_i + x_j) + \tau^2 x_i x_j + \delta_{ij}) \\ \mathrm{P}(AY_i) &= a^2 + s^2\sigma^2 + (ab + s^2\sigma\tau\rho)x_i \\ \mathrm{P}(BY_i) &= (ab + s^2\sigma\tau\rho) + (b^2 + s^2\tau^2)x_i \end{aligned}$$

We can now set up two belief structures, C *with base* $\{1, A, B\}$ *and* D *with base* $\{1, Y_1, \ldots, Y_n\}$.

We will use some example data sets to examine our methods, these are illustrated in Figure 3.1, and in which our prior beliefs are $s = 2$, $\sigma = 3$, $\tau = 2$, $\rho = -\frac{1}{2}$, $a = -5$ *and* $b = 5$. *For these parameter values our implied beliefs are*

$$
\begin{array}{lllllll}
\mathrm{P}(A) & = & -5 & \mathrm{P}(B) & = & 5 & \mathrm{P}(Y_i) = 5(x_i - 1) \\
\mathrm{P}(A^2) & = & 61 & \mathrm{P}(AB) & = & -37 & \mathrm{P}(AY_i) = 61 - 37x_i \\
 & & & \mathrm{P}(B^2) & = & 41 & \mathrm{P}(BY_i) = -37 + 41x_i \\
 & & & & & & \mathrm{P}(Y_iY_j) = 61 - 37(x_i + x_j) + \\
 & & & & & & \quad 41x_ix_j + 4\delta_{ij}
\end{array}
$$

3.4 Conditioning and projection

The notions in section 3.2 can be considered as projections. Let B be a closed subspace of the belief structure A, such that $b(B) = \{X_0, X_1, \ldots\}$, then the orthogonal projection of any general element $Y \in A$, denoted by $\mathrm{P}_B(Y)$ is the element $Y^* \in B$ that minimizes $\|Y - Y^*\|$. So letting $Y^* = (r_0, r_1, \ldots) = r_0X_0 + r_1X_1 + \cdots$, we need to choose $r_0, r_1, \ldots$ to minimize

$$\|Y - r_0X_0 - r_1X_1 - \cdots\|,$$

which is the same as minimizing L^*.

If B is not a subset of A, we can still define $\mathrm{P}_B(X)$, by first constructing the augmented belief structure $D = A + B$, then P_B can be considered to be the restriction of the operator $\mathrm{P}_B : D \to B$ to A, considering it as a subspace of of D.

If B has a finite base $b(B) = \{X_0, X_1, \ldots, X_n\}$, this operator can be easily evaluated with matrix operations . As the base of B is finite we can write $Y^* = (r_0, \ldots, r_n) = r_0X_0 + \cdots + r_nX_n$, and so to find Y^* we have to minimize

$$
\begin{aligned}
L^* &= \|Y - r_0X_0 - r_1X_1 - \cdots - r_nX_n\|^2 \\
&= \|Y\|^2 - 2\sum_{i=0}^{n} r_i(Y, X_i) + \sum_{i=0}^{n}\sum_{j=0}^{n} r_ir_j(X_i, X_j)
\end{aligned}
$$

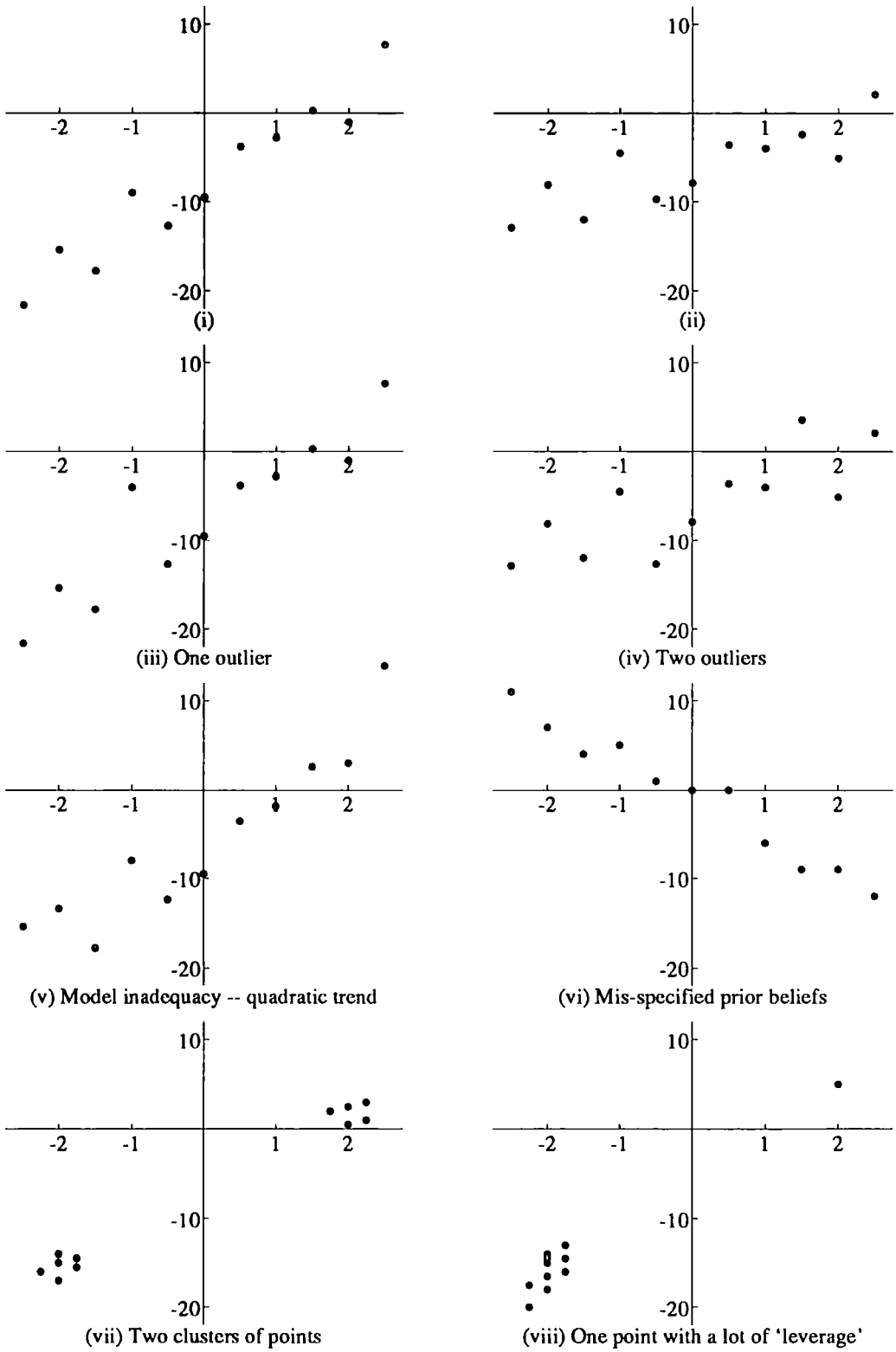

Figure 3.1: Data sets for regression example

Differentiating with respect to r_i and equating with zero gives

$$\frac{\partial L^*}{\partial r_i} = -2(X_i, X) + 2\sum_{j=0}^{n} r_j(X_i, X_j) = 0$$

and solving this gives in the vector form

$$Y^* = M_{BB}^{-1} m_{BA} \tag{3.2}$$

where

$$M_{BB} = [P(X_i X_j)]_{0\le i,j\le n} \quad \text{and} \quad m_{BA} = [P(X_0 Y), \ldots, P(X_n Y)]^{\mathrm{T}}$$

If A also has a finite base, $Y_0, \ldots, Y_m$, we can write $Y = (s_0, \ldots, s_m)^{\mathrm{T}} = s_0 Y_0 + \cdots + s_m Y_m$, (and putting $M_{BA} = [P(X_i Y_j)]_{0\le i\le n, 0\le j\le m}$) we obtain

$$\mathrm{P}_B(Y) = M_{BB}^{-1} M_{BA} Y \tag{3.3}$$

We can therefore represent P_B by a matrix, $P_B = M_{BB}^{-1} M_{BA}$.

Moreover, if we put

$$\begin{aligned} \mu_B &= [\mathrm{P}(X_1), \ldots, \mathrm{P}(X_n)]^{\mathrm{T}} & V_B &= [\mathrm{Cov}(X_i, X_j)]_{1\le i,j\le n} \\ \mu_A &= [\mathrm{P}(Y_1), \ldots, \mathrm{P}(Y_m)]^{\mathrm{T}} & C_{BA} &= [\mathrm{Cov}(X_i, Y_j)]_{1\le i\le n, 1\le j\le m} \end{aligned}$$

we can write

$$\begin{aligned} M_{BB} &= \begin{pmatrix} 1 & \mu_B{}^{\mathrm{T}} \\ \mu_B & \mu_B \mu_B{}^{\mathrm{T}} + V_B \end{pmatrix} \\ M_{BA} &= \begin{pmatrix} 1 & \mu_A{}^{\mathrm{T}} \\ \mu_B & \mu_B \mu_A{}^{\mathrm{T}} + C_{BA} \end{pmatrix} \end{aligned}$$

and then

$$P_B = \begin{pmatrix} 1 & \boldsymbol{\mu}_A{}^{\mathrm{T}} - \boldsymbol{\mu}_B{}^{\mathrm{T}} V_B^{-1} C_{BA} \\ \mathbf{0} & V_B^{-1} C_{BA} \end{pmatrix} \tag{3.4}$$

using the identity $\begin{pmatrix} 1 & \boldsymbol{b}^{\mathrm{T}} \\ \boldsymbol{b} & \boldsymbol{b}\boldsymbol{b}^{\mathrm{T}} + C \end{pmatrix}^{-1} = \begin{pmatrix} 1 + \boldsymbol{b}^{\mathrm{T}} C^{-1} \boldsymbol{b} & -\boldsymbol{b}^{\mathrm{T}} C^{-1} \\ C^{-1}\boldsymbol{b} & C^{-1} \end{pmatrix}$

In our example we can calculate P_D

$$P_D = \frac{1}{d} \begin{bmatrix} d & (1 + \sigma\tau\rho\Sigma_x + \tau^2\Sigma_{xx})a & -(n\sigma\tau\rho + \tau^2\Sigma_x)a \\ & -(\sigma^2\Sigma_x + \sigma\tau\rho\Sigma_{xx})b & +(1 + n\sigma^2 + \sigma\tau\rho\Sigma_x)b \\ \mathbf{0} & (\sigma^2 + \sigma^2\tau^2(1-\rho^2)\Sigma_{xx})\mathbf{1} & (\sigma\tau\rho - \sigma^2\tau^2(1-\rho^2)\Sigma_x)\mathbf{1} \\ & +(\sigma\tau\rho - \sigma^2\tau^2(1-\rho^2)\Sigma_x)\boldsymbol{x} & +(\tau^2 + n\sigma^2\tau^2(1-\rho^2))\boldsymbol{x} \end{bmatrix}$$

where $d = 1 + n\sigma^2 + 2\sigma\tau\rho\Sigma_x + \tau^2\Sigma_{xx} + \sigma^2\tau^2(1-\rho^2)(n\Sigma_{xx} - \Sigma_x^2)$

putting our prior beliefs in, gives

$$P_D = \begin{bmatrix} 1 & \frac{-5(1+6\Sigma_x+\Sigma_{xx})}{1+9n-6\Sigma_x+4\Sigma_{xx}+27(n\Sigma_{xx}-\Sigma_x^2)} & \frac{-5(1+6n+\Sigma_x)}{1+9n-6\Sigma_x+4\Sigma_{xx}+27(n\Sigma_{xx}-\Sigma_x^2)} \\ \mathbf{0} & \frac{9(1+3\Sigma_{xx})\mathbf{1}-3(1+9\Sigma_x)\boldsymbol{x}}{1+9n-6\Sigma_x+4\Sigma_{xx}+27(n\Sigma_{xx}-\Sigma_x^2)} & \frac{-3(1+9\Sigma_x)\mathbf{1}+9(4+27n)\boldsymbol{x}}{1+9n-6\Sigma_x+4\Sigma_{xx}+27(n\Sigma_{xx}-\Sigma_x^2)} \end{bmatrix}$$

For the data sets in Figure 3.1, we obtain the matrices in Table 3.1

Data set	P_D
1-6	$\begin{pmatrix} 1 & -0.0170 & 0.0400 \\ 0 & 0.08971 - 0.0004\boldsymbol{x} & -0.00041 + 0.0359\boldsymbol{x} \end{pmatrix}$
7	$\begin{pmatrix} 1 & -0.0136 & 0.0244 \\ 0 & 0.09001 + 0.0028\boldsymbol{x} & 0.00281 + 0.0225\boldsymbol{x} \end{pmatrix}$
8	$\begin{pmatrix} 1 & 0.0656 & 0.0520 \\ 0 & 0.24881 + 0.0503\boldsymbol{x} & 0.05031 + 0.0040\boldsymbol{x} \end{pmatrix}$

Table 3.1: P_D for data sets

To finish this section, it is useful to point out three properties of projections in general;

a) Projections are linear:

$$P_B(X + Y) = P_B(X) + P_B(Y)$$

In this context it just re-iterates that the notion of prevision is a linear property.

b) Projections are idempotent:

$$P_B(P_B(X)) = P_B(X)$$

Here it says you can get no more information out of a piece of data, if you re-use it later.

c) Projections are self-adjoint:

$$(X, P_B(Y)) = (P_B(X), Y)$$

3.5 Adjusting beliefs

In general if we revise our beliefs about some general random quantity Y of interest, we can re-assess our prevision about Y to get a new prevision $P^*(Y)$. Many ways can be used to update this prevision but we will concentrate on the most mechanical one here, for further reference see Goldstein[1983]. If we have constructed a belief structure B representing relevant data, one way of "updating" our prevision of Y is to evaluate the projection of Y on B, i.e.

$$P^*(Y) = P_B(Y)$$

Y can then be rewritten as $Y = (Y - P_B(Y)) + P_B(Y)$, these two terms are orthogonal so the variance of $Y = \|Y - P(Y)\|^2$ can be split up as follows

$$\mathrm{Var}(Y) = \mathrm{Var}(Y - P_B(Y)) + \mathrm{Var}(P_B(Y)). \tag{3.5}$$

The two variance terms are the residual variability of Y given the data, and the variability of Y that can be accounted for by the variability in the data.

Defn. 3.5 *We call the residual quantity* $Y - \mathrm{P}_B(Y)$ *the* adjusted *version of* Y *and denote it by* $[Y/B]$, *and as an intuitive measure of the information obtained we define*

$$D(Y) = \frac{\mathrm{Var}([Y/B])}{\mathrm{Var}(Y)}$$

the adjustment ratio *for* Y *induced by* B.

We also define the complimentary quantity $R(Y) = \frac{\mathrm{Var}(\mathrm{P}_B(Y))}{\mathrm{Var}(Y)} = 1 - D(Y)$. The measure $R(Y)$ is a measure of the explanatory power of B, if $R(Y)$ is nearly one then B contains quantities which explain Y well, but if $R(Y)$ is near zero, it contains quantities of low explanatory power, or possibly we have not chosen to specify useful aspects of the quantities.

If we use the notation in equation (3.4), we have

$$\begin{aligned} \mathrm{Var}(P_B(Y)) &= \boldsymbol{c}_{BA}^{\mathrm{T}} V_B^{-1} \boldsymbol{c}_{BA} \\ \mathrm{Var}[Y/B] &= v_A - \boldsymbol{c}_{BA}^{\mathrm{T}} V_B^{-1} \boldsymbol{c}_{BA} \end{aligned}$$

and

$$D(Y) = 1 - \boldsymbol{c}_{BA}^{\mathrm{T}} V_B^{-1} \boldsymbol{c}_{BA} v_A^{-1} \tag{3.6}$$

$$R(Y) = \boldsymbol{c}_{BA}^{\mathrm{T}} V_B^{-1} \boldsymbol{c}_{BA} v_A^{-1} \tag{3.7}$$

where $\boldsymbol{c}_{BA} = (\mathrm{Cov}(X_1, Y), \ldots, \mathrm{Cov}(X_n, Y))^{\mathrm{T}}$ and $v_A = \mathrm{Var}(Y)$

In our regression example we have

$$\begin{aligned} D(A) &= s^2\sigma^2 \frac{1 + \tau^2 \Sigma_{xx}(1 - \rho^2)}{d} \\ D(B) &= s^2\tau^2 \frac{1 + \sigma^2 n(1 - \rho^2)}{d} \end{aligned}$$

again incorporating our prior beliefs about A and B we have for our example data sets, values of D(A) and D(B) summarized in Table 3.2. We see here that we learn very little about A in the last example, as D(A) is near one.

Data set	$D(A)$	$D(B)$
1–6	0.3588	0.1437
7	0.3601	0.0898
8	0.9951	0.2541

Table 3.2: Values of $D(A)$ and $D(B)$ for example data sets.

3.6 Adjusted belief structures

We can extend this notion of adjusting beliefs to adjusting belief structures. When we are analysing our beliefs, our main concern is how a set of statements about observable data can alter our collection of belief statements. When we do a full Bayesian analysis we obtain a posterior distribution by updating our prior distribution with the likelihood, on the other hand with limited belief specification we are interested how one belief structure is modified by observing another belief structure.

Defn. 3.6 *If A and D are belief structures,* $[A/D]$ *is the* belief structure A adjusted for the belief structure D, *and is defined as the belief structure with base* $b([A/D]) = [b(A)/D]$. *Where for a set of random quantities* $C = \{Z_1, \ldots, Z_n\}$, $[C/D] = \{[Z_1/D], \ldots, [Z_n/D]\}$.

an alternative but equivalent definition, that does not need an explicit construction of bases is

Defn. 3.6* *If A and D are belief structures, then the* belief structure A adjusted for the belief structure D *is defined as the orthogonal complement* $D^{\perp}$ *of* D *in* $A + D$.

Now if we denote by I the belief structure whose base is just the unit constant, then $[A/I]$ is equivalent to the structure obtained if we use the alternative inner product $(X,Y)' = \mathrm{Cov}(X,Y)$ which de Finetti uses in his geometric interpretation in chapter 4 of his book *The Theory of Probability*[1970]. This at first seems a more natural inner product, quantities with large norms are those with high variability, and quantities that are orthogonal have zero correlation. However with this inner product all elements are standardized to have zero prevision, so the 'first order' information is lost if we just examine the inner product. This adjustment is useful as it splits our beliefs about A into two parts, our beliefs about the previsions of the quantities in A and their variability, and allows us to focus on the second of these which is of more interest to us. We can expand this to any adjustment by any belief structure, and we need to find ways of quantifying and using the residual variation.

We can now include some of the properties of adjusted belief structures

Property 3.1 $[A/D] = 0$ *if and only if* A *is contained in* D.

i.e. You can only get perfect information about the elements of A if it is contained in the data you are given.

Property 3.2 $[A/D] = A$ *if and only if* A *is orthogonal to* D.

i.e. Uncorrelated data gives no information at all.

Property 3.3 *For any* A_1,A_2 *and* D,

$$[(A_1 + A_2)/D] = [A_1/D] + [A_2/D].$$

Property 3.4 *For any* $A_1, \ldots, A_k$,

$$A_1 + A_2 + \cdots + A_k = D_1 + D_2 + \cdots + D_k,$$

where we define $D_1 = A_1$, *and, for* $i \geq 2$, $D_i = [A_i/(A_1 + \cdots + A_{i-1}]$, *and* $D_1, \ldots, D_k$ *are mutually orthogonal.*

Property 3.5 *For any A and any D_1 which is orthogonal to D_2,*

$$[A/(D_1 + D_2)] = [[A/D_1]/D_2] = [[A/D_2]/D_1],$$

and if we ***lift the restriction*** *that D_1 is orthogonal to D_2,*

$$[A/(D_1 + D_2)] = [[A/D_1]/[D_2/D_1]] = [[A/D_2]/[D_1/D_2]].$$

This last property is comparable to the conditional probability statement $\Pr(A|BC) = \frac{\Pr(AB|C)}{\Pr(B|C)}$, and in Bayesian statistics to using your posterior distribution from one stage, as a prior distribution for the next.

We can also obtain some relationships between projection and adjustment

Property 3.6 *For any B,D and any X in A,*

$$\mathrm{P}_{B+D}(X) = \mathrm{P}_B(X) + \mathrm{P}_{[D/B]}(X).$$

Property 3.7 *For any B,D and any X,Y in A,*

$$(P_{B+D}(X), P_{B+D}(Y)) = (\mathrm{P}_B(X), \mathrm{P}_B(Y)) + (\mathrm{P}_{[D/B]}(X), \mathrm{P}_{[D/B]}(Y))$$

In particular for any D,

$$\begin{aligned}(X, Y) &= (\mathrm{P}_D(X), \mathrm{P}_D(Y)) + (\mathrm{P}_{[A/D]}(X), \mathrm{P}_{[A/D]}(Y)) \\ &= (\mathrm{P}_D(X), \mathrm{P}_D(Y)) + ([X/D], [Y/D])\end{aligned}$$

This is the more general form of equation (3.5).

3.7 Diagnostics – bearing and length

Now we have defined the concept of belief structures, and indicated one (probably the simplest) way of adjusting them, we need to assign ways of measuring this adjustment.

Given we have assigned our present (time t) previsions for $C = \{X_1, \ldots, X_n\}$, $\mathrm{P}(C)$, and also our present covariance structure i.e. $\mathrm{P}(X_i X_j)$ for all X_i, X_j in C, we can produce an orthonormal (uncorrelated) basis for our inner product space.

Defn. 3.7 *A* Component Representation *of C is any set $\{E_1, \ldots, E_r\}$ of r linear combinations of $\{X_0, \ldots, X_n\}$, where $X_0 = 1$, such that*

a) $\mathrm{P}(E_i) = 0$

b) $\mathrm{P}(E_i E_j) = \delta_{ij}$

c) Each X_i in C can be expressed as a linear combination of $E_0 = 1$ and $E_1, \ldots, E_r$.

Now at a future time point t^*, we revise our beliefs about the elements of C, we define $\mathrm{P}^*(C)$, a new set of previsions. We now use the component representation obtained at time t to summarise the relative location of our new beliefs.

Defn. 3.8 *If $\{E_1, \ldots, E_r\}$ is a component representation of C, them the* bearing of $\mathrm{P}^*(C)$ with respect to $\mathrm{P}(C)$ *is*

$$Y^* = \mathrm{P}^*(E_1)E_1 + \cdots + \mathrm{P}^*(E_r)E_r$$

Lemma 3.1 *The bearing is independent of the choice of component representation.*

Proof

If we choose an alternate component representation $\{E'_1, \ldots, E'_r$, we can write this as $(E'_1, \ldots, E'_r)^{\mathrm{T}} = (QE_1, \ldots, QE_r)^{\mathrm{T}}$, where $Q^{\mathrm{T}}Q = I_r$, so

$$\begin{aligned} Y'^* &= \mathrm{P}^*((E'_1, \ldots, E'_r)^{\mathrm{T}})(E'_1, \ldots, E'_r) \\ &= \mathrm{P}^*((E_1, \ldots, E_r)^{\mathrm{T}})Q^{\mathrm{T}}Q(E_1, \ldots, E_r) \\ &= Y^* \end{aligned}$$

□

Lemma 3.2 *For all X in $\mathcal{L}_C$ we have*

$$\mathrm{P}^*(X) - \mathrm{P}(X) = \mathrm{Cov}(X, Y^*)$$

where the covariance is defined at time t

Proof

Every X in $\mathcal{L}_C$ can be expressed as

$$X = (X, E_0)E_0 + (X, E_1)E_1 + \cdots + (X, E_r)E_r$$

because $[E_0, \ldots, E_r]$ is an orthogonal basis for $\mathcal{L}_C$

$$X = \mathrm{P}(X) + \mathrm{Cov}(X, E_1)E_1 + \cdots + \mathrm{Cov}(X, E_r)E_r$$

as $\mathrm{P}(E_i) = 0$ for $i \neq 0$ then $\mathrm{Cov}(X, E_i) = \mathrm{P}(XE_i) - \mathrm{P}(X)\mathrm{P}(E_i) = (X, E_i)$, and $(X, E_0) = \mathrm{P}(X)$. So taking the prevision at time t^* we obtain

$$\begin{aligned}
\mathrm{P}^*(X) &= \mathrm{P}(X) + \mathrm{Cov}(X, E_1)\mathrm{P}^*(E_1) + \cdots + \mathrm{Cov}(X, E_r)\mathrm{P}^*(E_r) \\
&= \mathrm{P}(X) + \mathrm{Cov}(X, E_1 P^*(E_1) + \cdots + E_r \mathrm{P}^*(E_r)) \\
&= \mathrm{P}(X) + \mathrm{Cov}(X, Y^*)
\end{aligned}$$

□

It should be noted in particular, that if X is uncorrelated with the bearing then the prevision at time t^* is the same as the prevision at time t. So the bearing indicates the 'direction' of change of belief, it also shows the size of this change.

Defn. 3.9 *The* length *L^* of the revision of beliefs over $\mathcal{L}_C$ is defined by*

$$L^* = \sqrt{\mathrm{Var}(Y^*)} = \sqrt{(Y^*, Y^*)}$$

This length is related to the standardised change in belief by the following theorem

Corollary 3.3 *The maximum value of* $|\mathrm{P}^*(X) - \mathrm{P}(X)|/\sqrt{\mathrm{Var}(X)}$, *over all* X *in* $\mathcal{L}_C$ *is* L^*, *this maximum being obtained at* Y^*.

So in general if $Z^* = kY^*$, then a bearing of Z^* would represent a k-fold increase in the change of prevision for every element of $\mathcal{L}_C$. It should be noted that the bearing plays a similar role to the likelihood in a full Bayesian analysis.

In our example we can find the bearing given all the data.

$$Y_* = \frac{1}{s^2 d}(A-a \;\; B-b)\begin{bmatrix} 1+\sigma\tau\rho\Sigma_x+\tau^2\Sigma_{xx} & -\tau^2\Sigma_x-n\sigma\tau\rho \\ -\sigma^2\Sigma_x-\sigma\tau\rho\Sigma_{xx} & 1+\sigma\tau\rho\Sigma_x+n\sigma^2 \end{bmatrix}\begin{bmatrix} \Sigma_y-an-b\Sigma_x \\ \Sigma_{yx}-a\Sigma_x-b\Sigma_{xx} \end{bmatrix}$$

For the data sets we calculate the bearings and lengths listed in Table 3.3.

Data set	Bearing - Y^*	$\mathrm{Var}(Y^*)$	L^*
1	$-0.1176A - 0.0927B - 0.1245$	0.3736	0.6113
2	$-0.1368A - 0.2836B + 0.7337$	1.0293	1.0146
3	$-0.1059A - 0.0953B - 0.0532$	0.3071	0.5542
4	$-0.1165A - 0.2448B + 0.6415$	0.7633	0.8737
5	$-0.0331A - 0.0131B - 0.0990$	0.0317	0.1781
6	$-0.0984A - 0.6542B + 3.7791$	5.6515	2.3773
7	$-0.0921A - 0.1152B + 0.1150$	0.2630	0.5128
8	$-0.0068A + 0.0147B - 0.1702$	0.0075	0.0864

Table 3.3: Bearings and Lengths for data sets

Data set six automatically stands out as being "abnormal" as its length is twice as high as the rest, but we do not notice anything unusual about the rest of the data sets, we therefore need to look at the revision more closely to pick up other features in the data.

3.8 Belief transforms

It would be useful to have an analogue to the quantities $D(Y)$ and $R(Y)$ defined in Section 3.5, to measure the magnitude of the revision we make when we adjust one belief structure, by another, rather than just one random quantity by a belief structure. We can summarise this by the belief transform,

Defn. 3.10 *The* belief transform over B induced by D *is the linear operator over* B *defined by*

$$T_D = \mathrm{P}_B \mathrm{P}_D : B \rightarrow B$$

where P_B, *and* P_D *are the orthogonal projections from* D *to* B *and from* B *to* D *respectively.*

We also define a complementary transform

Defn. 3.11 *The* complementary belief transform *of* T_D *is* $S_D = I - T_D$

These transforms have the following properties:

Property 3.8 S_D *and* T_D *are self-adjoint with respect to the inner product* $(\cdot, \cdot)$, *i.e.*

$$\begin{aligned} (X, T_D(Y)) &= (T_D(X), Y) \\ (X, S_D(Y)) &= (S_D(X), Y) \end{aligned}$$

Property 3.9 S_D *and* T_D *are linear, i.e.*

$$T_D(X + Y) = T_D(X) + T_D(Y)$$

Property 3.10 *The norms of the* S_D *and* T_D *are not greater than one.*

Property 3.11

$$(X, T_D(Y)) = (\mathrm{P}_D(X), \mathrm{P}_D(Y))$$

$$(X, S_D(Y)) = ([X/D], [Y/D])$$

Property 3.12 *S_D and T_D decompose the inner product*

$$(X, Y) = (X, T_D(Y)) + (X, S_D(Y))$$

Property 3.13 ***Belief transforms*** *can be constructed sequentially, if D_1 and D_2 are belief structures, and using Properties 3.11, 3.5 and 3.7*

$$T_{(D_1+D_2)} = T_{D_1} + T_{[D_2/D_1]}$$

where $T_{[D_2/D_1]}$ is the belief transform over $[B/D_1]$ induced by $[D_2/D_1]$, but thought of as an operator over B.

Property 3.14 *If we choose D_2 so that we have $A = D_1 + D_2$, we have*

$$I = T_A = T_{D_1} + T_{[A/D_1]}$$

So

$$S_{D_1} = T_{[A/D_1]}$$

If we consider adjustment to be replacing the inner product on B by a new inner product $(X, Y)^* = (X - \mathrm{P}_D(X), Y - \mathrm{P}_D(Y))$, then T_D and S_D have the following properties;

Property 3.15

$$(X, Y)^* = (X, S_D(Y)) = (S_D(X), Y)$$

Property 3.16 *For any X in $\mathcal{L}$,*

$$D(X) = \frac{(X, S_D(X))}{(X, X)} \quad \text{and} \quad R(X) = \frac{(X, T_D(X))}{(X, X)}$$

Again if the bases of B and D are finite, the transforms T_D and S_D can be represented by the matrices (using the same notation as section 3.4)

$$T_D = M_{BB}^{-1} M_{BD} M_{DD}^{-1} M_{DB} \quad \text{and} \quad S_D = I - M_{BB}^{-1} M_{BD} M_{DD}^{-1} M_{DB} \tag{3.8}$$

respectively. Expanding these in terms of the first-order previsions and second-order covariance structure gives us

$$T_D = \begin{pmatrix} 1 & \boldsymbol{\mu}_B^{\mathrm{T}}(I - V_B^{-1} C_{BD} V_D^{-1} C_{DB}) \\ \mathbf{0} & V_B^{-1} C_{BD} V_D^{-1} C_{DB} \end{pmatrix} \tag{3.9}$$

$$S_D = \begin{pmatrix} 0 & \boldsymbol{\mu}_B^{\mathrm{T}}(V_B^{-1} C_{BD} V_D^{-1} C_{DB} - I) \\ \mathbf{0} & I - V_B^{-1} C_{BD} V_D^{-1} C_{DB} \end{pmatrix} \tag{3.10}$$

In the regression example we have the two transforms

$$T_D = \frac{1}{d} \begin{bmatrix} d & [(1 + \sigma\tau\rho\Sigma_x + \tau^2\Sigma_{xx})a - (\sigma^2\Sigma_x + \sigma\tau\rho\Sigma_{xx})b]\mathbf{1}^{\mathrm{T}} \\ & +[-(n\sigma\tau\rho + \Sigma_x\tau^2)a + (1 + n\sigma^2 + \sigma\tau\rho\Sigma_x)b]\boldsymbol{x}^{\mathrm{T}} \\ \mathbf{0} & (\sigma^2 + \sigma^2\tau^2(1-\rho^2)\Sigma_{xx})\mathbf{1}\mathbf{1}^{\mathrm{T}} + (\tau^2 + n\tau^2\sigma^2(1-\rho^2))\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}} \\ & +(\sigma\tau\rho - \sigma^2\tau^2(1-\rho^2)\Sigma_x)(\mathbf{1}\boldsymbol{x}^{\mathrm{T}} + \boldsymbol{x}\mathbf{1}^{\mathrm{T}}) \end{bmatrix}$$

$$T_C = \frac{1}{d} \begin{bmatrix} d & (1 + \sigma\tau\rho\Sigma_x + \tau^2\Sigma_{xx})a & -(n\sigma\tau\rho + \tau^2\Sigma_x)a \\ & -(\sigma^2\Sigma_x + \sigma\tau\rho\Sigma_{xx})b & +(1 + n\sigma^2 + \sigma\tau\rho\Sigma_x)b \\ 0 & n\sigma^2 + \sigma\tau\rho\Sigma_x + & n\sigma\tau\rho + \tau^2\Sigma_x \\ & \sigma^2\tau^2(1-\rho^2)(n\Sigma_{xx} - \Sigma_x^2) & \\ 0 & \sigma^2\Sigma_x + \sigma\tau\rho\Sigma_{xx} & \sigma\tau\rho\Sigma_x + \tau^2\Sigma_{xx} + \\ & & \sigma^2\tau^2(1-\rho^2)(n\Sigma_{xx} - \Sigma_x^2) \end{bmatrix}$$

and for the particular data sets we have

$$
\begin{aligned}
&\textit{Data sets 1–6}\\
T_D &= \begin{bmatrix} 1 & -0.01701 + 0.0400\boldsymbol{x} \\ \mathbf{0} & 0.897011^{\mathrm{T}} - 0.0004(\boldsymbol{x}1^{\mathrm{T}} + 1\boldsymbol{x}^{\mathrm{T}}) + 0.0359\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}} \end{bmatrix}\\
T_C &= \begin{bmatrix} 1 & -0.0170 & 0.0400 \\ 0 & 0.9868 & -0.0039 \\ 0 & -0.0098 & 0.9881 \end{bmatrix}\\
&\textit{Data set 7}\\
T_D &= \begin{bmatrix} 1 & -0.01361 + 0.0244\boldsymbol{x} \\ \mathbf{0} & 0.900511^{\mathrm{T}} - 0.0028(\boldsymbol{x}1^{\mathrm{T}} + 1\boldsymbol{x}^{\mathrm{T}}) + 0.0225\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}} \end{bmatrix}\\
T_C &= \begin{bmatrix} 1 & -0.0136 & 0.0244 \\ 0 & 0.9863 & -0.0030 \\ 0 & -0.0110 & 0.9922 \end{bmatrix}\\
&\textit{Data set 8}\\
T_D &= \begin{bmatrix} 1 & -0.06561 + 0.0520\boldsymbol{x} \\ \mathbf{0} & 0.248811^{\mathrm{T}} - 0.1005(\boldsymbol{x}1^{\mathrm{T}} + 1\boldsymbol{x}^{\mathrm{T}}) + 0.0635\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}} \end{bmatrix}\\
T_C &= \begin{bmatrix} 1 & -0.0656 & 0.0520 \\ 0 & 0.9520 & -0.0220 \\ 0 & -0.0612 & 0.9676 \end{bmatrix}
\end{aligned}
$$

It is relatively difficult to understand the complexities of the transform just by looking at it, and so we need to obtain a summary of it similar in nature to the summaries $D(X)$ and $R(X)$ we obtained when we adjusted one random quantity by a belief structure, and to do this it is useful to analyse the eigenstructure of these transforms. We will assume here that the base of B is finite, and T_D has a set of orthonormal eigenvectors $E_0 = 1, E_1, \ldots, E_r$, corresponding to the eigenvalues $1 = \lambda_0 \geq \lambda_1 \geq \cdots \geq \lambda_r \geq 0$. The set $\{E_1, \ldots, E_r\}$ is a component represention, and so its elements are uncorrelated with zero mean and unit variance, and we have $\mathrm{P}_B\mathrm{P}_D(E_i) = \lambda_i E_i$.

Defn. 3.12 *If we let* $M(B) = \{E_1, \ldots, E_s\}$ *be the subset of* $\{E_1, \ldots, E_r\}$ *which have non-zero eigenvalues, then we call* $M(B)$ *a* map over B.

Defn. 3.13 *We call the values* $\lambda_1 \geq \cdots \geq \lambda_s$ *the* scale *of the map.*

As $R(E_i) = \lambda_i$, axes of the map which have a large eigenvalue are those where a large expected change in belief will occur, and those that correspond to a small change in belief have correspondingly small eigenvalues. Finally, those eigenvectors with zero eigenvalues correspond to directions in which the data will give no information at all.

We can also consider the complementary transform S_D which has the same eigenvectors, but with eigenvalues $\nu_i = 1 - \lambda_i$. As B and D are both belief structures we can also consider the transforms T_B and S_B. It can be shown that the non-zero eigenvalues of T_B are precisely the non-zero eigenvalues of T_D, and similarly the eigenvalues of S_B less than one are the same as for S_D. Also the eigenvectors (corresponding to non-zero eigenvalues) of T_D and T_B are related as follows

Lemma 3.4 *For each* i *in* $\{0, \ldots, s\}$, $F_i = \frac{1}{\sqrt{\lambda_i}} \mathrm{P}_D(E_i)$ *is the eigenvector of* T_B *with unit variance corresponding to the eigenvalue* λ_i. *Therefore* $\{F_1, \ldots, F_s\}$ *is a map,* $M(D)$, *over* D *induced by* B, *with the same scale as* $M(B)$

Note: This result can be seen to be useful from a practical point of view, instead of computing what could be a large belief transform T_D, it is possible to compute a smaller one T_B, whose eigenvalues and eigenvectors are easier to compute, and then making the corresponding transformation. For example, if you were examining how one data point influenced your beliefs about a set of n quantities, the eigenanalysis of the 2-by-2 belief transform T_B is easier to perform than the eigenanalysis of the $(n+1)^2$ belief transform T_D.

Defn. 3.14 *The maps* $M(B)$ *and* $M(D)$ *defined as above are called the* twin maps *for the pair of belief structures* B *and* D.

If we now perform an eigenanalysis on the transforms in our example, we can get their scales and maps

If $\tau^2\Sigma_x = -n\rho\sigma\tau$

$$\begin{aligned}
\lambda_1 &= \frac{n(1-\rho^2)\sigma^2}{1+n(1-\rho^2)\sigma^2} \\
\lambda_2 &= \frac{\tau^2\Sigma_{xx} - n\rho^2\sigma^2}{1+\tau^2\Sigma_{xx} - n\rho^2\sigma^2} \\
M(C) &= \{\frac{\tau^2(A-a) - \sigma\tau\rho(B-b)}{s\tau^2\sigma\sqrt{1-\rho^2}}, \frac{B-b}{s\tau}\}
\end{aligned}$$

If $\sigma^2\Sigma_x = -\Sigma_{xx}\rho\sigma\tau$

$$\begin{aligned}
\lambda_1 &= \frac{n\sigma^2 - \rho^2\tau^2\Sigma_{xx}}{1+n\sigma^2 - \rho^2\tau^2\Sigma_{xx}} \\
\lambda_2 &= \frac{\tau^2\Sigma_{xx}(1-\rho^2)}{1+\tau^2\Sigma_{xx}(1-\rho^2)} \\
M(C) &= \{\frac{A-a}{s\sigma}, \frac{\sigma^2(B-b) - \sigma\tau\rho(A-a)}{s\tau\sigma^2\sqrt{1-\rho^2}}\}
\end{aligned}$$

Otherwise

$$\begin{aligned}
\lambda_1 &= (2d)^{-1}\Big(n\sigma^2 + \tau^2\Sigma_{xx} + 2\sigma\tau\rho\Sigma_x + 2\sigma^2\tau^2(1-\rho^2)(n\Sigma_{xx} - \Sigma_x^2) + \\
&\qquad \sqrt{(n\sigma^2 - \tau^2\Sigma_{xx})^2 + 4(n\sigma\tau\rho + \tau^2\Sigma_x)(\sigma^2\Sigma_x + \sigma\tau\rho\Sigma_{xx})}\Big) \\
\lambda_2 &= (2d)^{-1}\Big(n\sigma^2 + \tau^2\Sigma_{xx} + 2\sigma\tau\rho\Sigma_x + 2\sigma^2\tau^2(1-\rho^2)(n\Sigma_{xx} - \Sigma_x^2) - \\
&\qquad \sqrt{(n\sigma^2 - \tau^2\Sigma_{xx})^2 + 4(n\sigma\tau\rho + \tau^2\Sigma_x)(\sigma^2\Sigma_x + \sigma\tau\rho\Sigma_{xx})}\Big) \\
\boldsymbol{e}_1 &\propto 2(n\sigma\tau\rho + \tau^2\Sigma_x)(A-a) + \Big(\tau^2\Sigma_{xx} - n\sigma^2 + \\
&\qquad \sqrt{(n\sigma^2 - \tau^2\Sigma_{xx})^2 + 4(n\sigma\tau\rho + \tau^2\Sigma_x)(\sigma^2\Sigma_x + \sigma\tau\rho\Sigma_{xx})}\Big)(B-b) \\
\boldsymbol{e}_2 &\propto 2(n\sigma\tau\rho + \tau^2\Sigma_x)(A-a) + \Big(\tau^2\Sigma_{xx} - n\sigma^2 - \\
&\qquad \sqrt{(n\sigma^2 - \tau^2\Sigma_{xx})^2 + 4(n\sigma\tau\rho + \tau^2\Sigma_x)(\sigma^2\Sigma_x + \sigma\tau\rho\Sigma_{xx})}\Big)(B-b)
\end{aligned}$$

If $\tau^2\Sigma_x = -n\rho\sigma\tau$

In our particular example sets we have the eigenvalues listed in Table 3.4, the eigenvalues for the first seven data sets are similar, but for the eighth, the second eigenvalue is smaller, this comes from the assymetry of

the design.

Data set	λ_1	λ_2
1–6	0.9937	0.9811
7	0.9956	0.9829
8	0.9972	0.9223

Table 3.4: Eigenvalues of belief transforms in regression example

The bearing can be evaluated using these maps in a symmetric way.

Lemma 3.5 *If* $M(B) = \{E_1, \ldots, E_s\}$ *and* $M(D) = \{F_1, \ldots, F_s\}$ *the maps defined above with scale* $\{\lambda_1 \geq \ldots \geq \lambda_t\}$. *Then the bearing over* B *induced by* D *and over* D *induced by* B *both have the form*

$$\sqrt{\lambda_1} E_1 F_1 + \cdots + \sqrt{\lambda_s} E_s F_s$$

where if we observe $D = d$, Y_d *is calculated by putting* $F_i = f_i$, *but if we observe* $B = b$, Y_b *is calculated by putting* $E_i = e_i$.

Corollary 3.6

$$\mathrm{P}(\mathrm{Var}(Y_D)) = \mathrm{trace}(T_D) - 1$$

Proof

If we observe $D = d$, $\mathrm{Var}(Y_D) = \lambda_1 f_1^2 + \cdots + \lambda_s f_s^2$, and the prevision of this is $\mathrm{P}(\lambda_1 F_1^2 + \cdots + \lambda_s F_s^2) = \lambda_1 P(F_1^2) + \cdots + \lambda_s P(F_s^2) = \lambda_1 + \cdots + \lambda_s = \mathrm{trace}(T_D) - 1$. □

So in our example, the trace of the transform is

$$\mathrm{trace}(T_D) - 1 = 1 + \frac{\sigma^2 \tau^2 (1 - \rho^2)(n\Sigma_{xx} - \Sigma_x^2) - 1}{d}$$

So we now have a simple summary to alert us to anything unexpected in the data, we can compute the value of $\mathrm{Var}(Y_d)$ and compare it with the trace of the transform, if it is large then the data is exhibiting anomalous behaviour.

In our example for data sets 1-6 the trace is 1.9748; data set 7, 1.9786; and data set 8 is 1.9176.

These maps have another use other than calculating the bearing, for if we make the following definition

Defn. 3.15 *We define the* heart *of the transform T_D, $H[D/B]$, as the belief structure with base $[F_1, \ldots, F_s]$.*

we see the heart of T_D, $H[D/B]$, corresponds to the dual map $M(D)$, of T_D, and has the following useful properties:

Property 3.17 *$H[D/B]$ is contained in $[D/B]$.*

Property 3.18 *For all X in B,*

$$T_D(X) = T_{H[D/B]}(X).$$

Property 3.19 *If $H[D/B_1] = \cdots = H[D/B_m] = H$ then $H[D/(B_1 + \cdots + B_m)] = H$ also.*

The heart therefore summarizes exactly which aspects of the data influence the revision of belief. For example if we are looking at the mean of a random quantity, and the belief structure D consists of a set of independent observations of that quantity, then the only part of D that gives us any information about the mean is the sample mean, so the heart of the transform is simply the sample mean. Property 3.19 simply says if all the revision of belief structures $B_1, \ldots, B_m$ only depend on certain *sufficient statistics* then the adjustment of the combined belief structure $B_1 + \cdots + B_m$ only depends on the set. More generally we have,

Property 3.20

$$H[D/(B_1 + \cdots + B_m)] = H[D/B_1] + \cdots + H[D/B_m]$$

Chapter 4

Grid Based Design Criteria

We will now use the results from the previous chapter to construct designs for the interpolation problem (using Currin *at al*'s[1991] model) from Chapter 2.

We will assume that we wish to make inference about the function on some grid $G = \{x_1, \ldots\}$, by making function evaluations at a set of design points $\mathcal{X} = \{x_{(1)}, \ldots, x_{(n)}\} \subset G$. We will use as a criteria the trace of the belief transform, which is a descriptive tool measuring how much information we can receive by taking function evaluations at the design points. If we look at the transform of G by $\mathcal{X}$, we will note that this always has trace n, because we reduce the variance of $Y(\cdot)$ at $x_{(1)}$, $x_{(2)}, \ldots, x_{(n)}$ by a factor of 1 each. As we know we have perfect information about the function at the design points we are interested in what we can learn about the function elsewhere, we could therefore look at the transform $T_{\mathcal{X}}$ of $G - \mathcal{X}$ by $\mathcal{X}$, and find its trace. To find this directly is straight forward, but difficult to do computationally, but it can be simplified by using Theorem 4.1, and its corollary.

Theorem 4.1 *The transform $T_{(G-\mathcal{X})}$ of $\mathcal{X}$ can be found as follows*

$$T_{(G-\mathcal{X})} = I - (BV_{\mathcal{X}\mathcal{X}})^{-1} \tag{4.1}$$

where

$$\begin{aligned}\mathbf{Var}(\boldsymbol{Y}) &= \begin{pmatrix} V_{(G-\mathcal{X})(G-\mathcal{X})} & V_{(G-\mathcal{X})\mathcal{X}} \\ V_{\mathcal{X}(G-\mathcal{X})} & V_{\mathcal{X}\mathcal{X}} \end{pmatrix} \\ \mathbf{Var}^{-1}(\boldsymbol{Y}) &= \begin{pmatrix} A & Z \\ Z^T & B \end{pmatrix}\end{aligned}$$

where $\boldsymbol{Y} = (Y(G-\mathcal{X}), Y(\mathcal{X}))^T$

Proof

If we invert $\mathrm{Var}(\boldsymbol{Y})$ we get

$$\begin{aligned}\begin{bmatrix} V_{(G-\mathcal{X})(G-\mathcal{X})} & V_{(G-\mathcal{X})\mathcal{X}} \\ V_{\mathcal{X}(G-\mathcal{X})} & V_{\mathcal{X}\mathcal{X}} \end{bmatrix}\begin{bmatrix} A & Z \\ Z^T & B \end{bmatrix} &= \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \\ Z &= -V^{-1}_{(G-\mathcal{X})(G-\mathcal{X})}V_{(G-\mathcal{X})\mathcal{X}}B \\ -V_{\mathcal{X}(G-\mathcal{X})}V^{-1}_{(G-\mathcal{X})(G-\mathcal{X})}V_{(G-\mathcal{X})\mathcal{X}}B + V_{\mathcal{X}\mathcal{X}}B &= I \\ -V^{-1}_{\mathcal{X}\mathcal{X}}V_{\mathcal{X}(G-\mathcal{X})}V^{-1}_{(G-\mathcal{X})(G-\mathcal{X})}V_{(G-\mathcal{X})\mathcal{X}} + I &= V^{-1}_{\mathcal{X}\mathcal{X}}B^{-1} \\ I - V^{-1}_{\mathcal{X}\mathcal{X}}B^{-1} &= T_{(G-\mathcal{X})}\end{aligned}$$

□

Corollary 4.2

$$\mathrm{Trace}T_{\mathcal{X}} = n - \mathrm{Trace}((BV_{\mathcal{X}\mathcal{X}})^{-1}) \tag{4.2}$$

We now use this result as a criterion for finding optimal grid based designs. To do this we first compute the inverse of the variance matrix for all the points in the grid (as we only need to do this once we can use high precision computer programs to invert the matrix), and then we can compute the trace for all the designs quickly, only needing to invert a $n \times n$ matrix each time. This criteria is similar to the entropy criteria in Currin *et al*[1991], and the designs have similarities, which we will observe later.

In the examples that follow we will use the correlation function $\rho_1(\cdot)$, as this allows us to find an analytic inverse for V.

4.1 Example 1 – $\rho_1(d)$ on a "Uniform" grid

We will start with a uniform grid as this is the easiest one to define, with $N+1$ points in the interval $[-1,1]$, $G = \{-1+2i/N : 0 \leq i \leq N\}$. $\mathrm{Var}(\boldsymbol{Y})$ and its inverse, have simple forms,

$$\mathrm{Var}(\boldsymbol{Y}) = \sigma^2 \begin{pmatrix} 1 & D & D^2 & \cdots & D^{N-1} & D^N \\ D & 1 & D & \cdots & D^{N-2} & D^{N-1} \\ D^2 & D & 1 & \cdots & D^{N-3} & D^{N-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ D^{N-1} & D^{N-2} & D^{N-3} & \cdots & 1 & D \\ D^N & D^{N-1} & D^{N-2} & \cdots & D & 1 \end{pmatrix} \tag{4.3}$$

$$\mathrm{Var}^{-1}(\boldsymbol{Y}) = \frac{1}{\sigma^2(1-D^2)} \begin{pmatrix} 1 & -D & 0 & \cdots & 0 \\ -D & 1+D^2 & -D & \ddots & \vdots \\ 0 & -D & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 1+D^2 & -D \\ 0 & \cdots & 0 & -D & 1 \end{pmatrix} \tag{4.4}$$

where $D = e^{-2\theta/N}$

We can then find the trace of the transform by partitioning the matrix for every set of design points to be considered, as an example let us assume we have the n design points $0 < i_1 < \ldots < i_n < N$, no two being adjacent, then $BV_{\mathcal{X}\mathcal{X}}$ becomes

$$(BV_{\mathcal{X}\mathcal{X}})_{jk} = \frac{1+D^2}{1-D^2} D^{|i_j - i_k|} \tag{4.5}$$

We can find its inverse analytically once more,

$$((BV_{\mathcal{X}\mathcal{X}})^{-1})_{jk} = \frac{1-D^2}{1+D^2}\begin{cases} \frac{-D^{i_k-i_j}}{1-D^{2(i_k-i_j)}} & k=j+1, \\ \frac{-D^{i_j-i_k}}{1-D^{2(i_j-i_k)}} & j=k+1, \\ \frac{1}{1-D^{2(i_2-i_1)}} & j=k=1, \\ \frac{1-D^{2(i_{j+1}-i_{j-1})}}{(1-D^{2(i_j-i_{j-1})})(1-D^{2(i_{j+1}-i_j)})} & 1<j=k<n, \\ \frac{1}{1-D^{2(i_n-i_{n-1})}} & j=k=n, \\ 0 & \text{otherwise.} \end{cases} \tag{4.6}$$

Which has trace

$$\text{Trace}((BV_{\mathcal{X}\mathcal{X}})^{-1}) = \frac{1-D^2}{1+D^2}\left[2-n+2\sum_{j=1}^{n-1}\frac{1}{1-D^{2(i_{j+1}-i_j)}}\right] \tag{4.7}$$

We are required to maximize $n - Tr((BV_{\mathcal{X}\mathcal{X}})^{-1})$, the maxima occur when the design points are as far away as possible from each other. For example for a four point design, on a grid of 33 points, we have the optimal design $\mathcal{X} = \{-\frac{15}{16}, -\frac{5}{16}, \frac{5}{16}, \frac{15}{16}\}$. The design never chooses the end points of the region in preference to interior points (as they give less information) because they have less neighbours. As we can analytically invert $BV_{\mathcal{X}\mathcal{X}}$, we can look at asymptotic results, letting the number of grid points tend to infinity, the optimal $m+1$ point design is $\mathcal{X} = \{-1+2\frac{i}{m} : 0 \le i \le m\}$.

4.2 Example 2 – $\rho_1(d)$ on a "Normal" grid

We repeat the previous example, but replace the grid by a non-uniform grid, the density of which is "normal". We will use a 21 point grid, G_* given by,

±1.9808	±1.4652	±1.1798	±0.9674
±0.7916	±0.6375	±0.4972	±0.3661
±0.2420	±0.1196	0	

We can view the grids here as discretizations of the metric $\Omega(\cdot)$ mentioned in Chapter 2, by choosing points which have equal probability separations, i.e. $\Omega(x_{i+1}) - \Omega(x_i)$ is a constant. The uniform grid in example 1 is a discretization of $\Omega(x) = 1/2,\ x \in [-1, 1]$, and the grid in this example is a discretization of the standard normal metric.

If we look at the criterion, we can see that the inverses of the two matrices $V_{\mathcal{X}\mathcal{X}}$ and B contribute different factors, the first is a measure of distance between the points in the grid, and the second is a measure of the density of the grid at the design points, and so the criterion is a compromise between the two, the former pushes the design points further out, and the latter pulling them back in.

In this case we will look for the designs for various values of θ ranging from 0.1 to 10, and for 1 to 4 design points, see Table 4.1.

Points	θ	Trace	Design		
1	0.1	0.9880357609	0.000000		
1	0.5	0.9402472164	0.000000		
1	1.0	0.8809195969	0.000000		
1	2.0	0.7651691290	0.000000		
1	10.0	0.1674219120	0.000000		
2	0.1	1.8801351469	±0.967422		
2	0.5	1.7965081434	±0.637484		
2	1.0	1.6878642412	±0.497201		
2	2.0	1.4705679971	±0.366106		
2	10.0	0.3204291940	-0.241040	0.366106	
3	0.1	2.6664274815	±1.179761	0.000000	
3	0.5	2.5576879674	±0.967422	0.000000	
3	1.0	2.4096985757	±0.791639	0.000000	
3	2.0	2.1058135644	±0.637484	0.000000	
3	10.0	0.4539349485	±0.241040	0.000000	
4	0.1	3.3438614935	±1.179761	±0.241040	
4	0.5	3.2173697441	±1.179761	±0.241040	
4	1.0	3.0329182492	±0.967422	-0.366106	0.241040
4	2.0	2.6616004396	±0.791639	±0.241040	
4	10.0	0.5780472657	±0.366106	±0.119648	

Table 4.1: 1-dimensional designs on a "normal" grid

4.3 Example 3 – Multi-dimensional grids

We can extend this method to allow $\boldsymbol{x}$ to be a m-dimensional vector. If we chose a product form for the covariance of the observations, and a lattice for the grid, then we note the variance matrix becomes a Krönecker product, of the "marginal variance matrices", and so the matrix inverse is simpler to compute. We can apply Theorem 4.1 and its corollary again, which makes a more dramatic saving of computing power, than in the one-dimensional case - the transform would usually be difficult to compute as it would require the inversion of a $(g^m - n) \times (g^m - n)$ and a $n \times n$ matrix for each calculation of the transform - but using this result we only need to invert m $g \times g$ matrices once to find B, and then an $n \times n$ matrix for each calculation of the transform.

Using the product correlation function $\rho(\boldsymbol{d}) = \prod_{i=1}^{m} e^{-|d_i|}$, we can generate optimal designs for higher dimensions. (The designs may be sub-optimal because they are generated by the following algorithm, the best of ten or more runs is used, which only finds locally optimal solutions.)

Choose a design at random

Repeat

For each point in turn

Choose the point to maximize the criterion

Until no more updating can be done.

We will produce examples for two and three dimensions on two 21^m point lattices, for one to ten points, these are summarised in the following series of tables and figures.

Table	Figure	Dimension	Grid
4.2	4.1&4.2	2	"Normal"
4.3	4.3&4.4	2	"Uniform"
4.4		3	"Normal"
4.5		3	"Uniform"

Where "Normal" refers to the grid from Example 2, and "Uniform" refers to the 21-point grid $\{-1.0, -0.9, \ldots, 0.9, 1.0\}$. Although not shown here the optimal designs

in the case of multidimensional "Uniform" girds are generally dependent on the value of θ, unlike there one-dimensional cousins.

Points	Trace(T)	Design		
1	0.985820	(0.0000, 0.0000)		
2	1.965840	(-0.3661,-0.3661)	(0.2410, 0.3661)	
3	2.943155	(-0.4972,-0.4972)	(0.4972,-0.1196)	(-0.1196, 0.4972)
4	3.918860	(0.1196, 0.6374)	(0.6374,-0.1196)	(-0.1196,-0.6374)
		(-0.6374, 0.1196)		
5	4.889073	(-0.1196,-0.6374)	(-0.6374, 0.0000)	(-0.2410, 0.7916)
		(0.7916,-0.2410)	(0.2410, 0.2410)	
6	5.857528	(-0.3661,-0.7916)	(0.7916, 0.2410)	(-0.7916, 0.3661)
		(0.1196, 0.7916)	(-0.1196, 0.0000)	(0.3661,-0.3661)
7	6.825350	(-0.4972,-0.4972)	(0.4972, 0.4972)	(-0.7916, 0.2410)
		(0.2410,-0.7916)	(0.7916,-0.2410)	(-0.2410, 0.7916)
		(0.0000, 0.0000)		
8	7.788608	(0.2410, 0.7916)	(-0.3661,-0.4972)	(-0.4972, 0.4972)
		(0.4972,-0.2410)	(0.1196,-0.9674)	(0.0000, 0.1196)
		(-0.9674,-0.1196)	(0.7916, 0.3661)	
9	8.751282	(0.4972, 0.3661)	(-0.4972,-0.3661)	(0.3661,-0.4972)
		(-0.3661, 0.4972)	(-0.1196,-0.9674)	(0.1196, 0.9674)
		(-0.9674, 0.1196)	(0.9674,-0.1196)	(0.0000, 0.0000)
10	9.710879	(0.0000,-0.3661)	(0.9674,-0.1196)	(-0.3661, 0.0000)
		(0.2410, 0.2410)	(-0.2410,-0.9674)	(0.6374, 0.6374)
		(-0.1196, 0.9674)	(-0.6374, 0.4972)	(-0.9674,-0.2410)
		(0.4972,-0.6374)		

Table 4.2: 2-dimensional designs on a "normal" grid

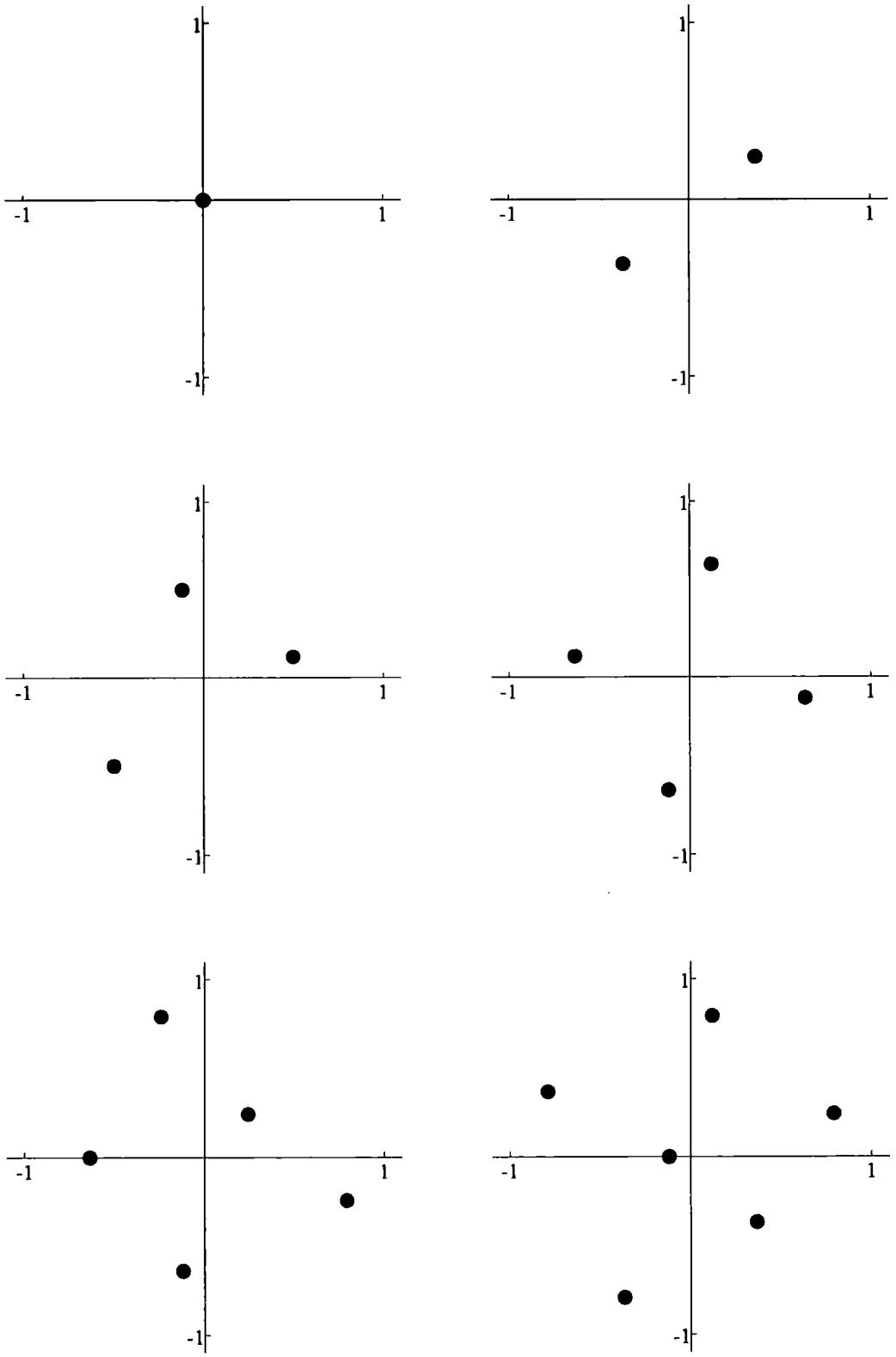

Figure 4.1: 2-dimensional designs on a "normal" grid with one to six points

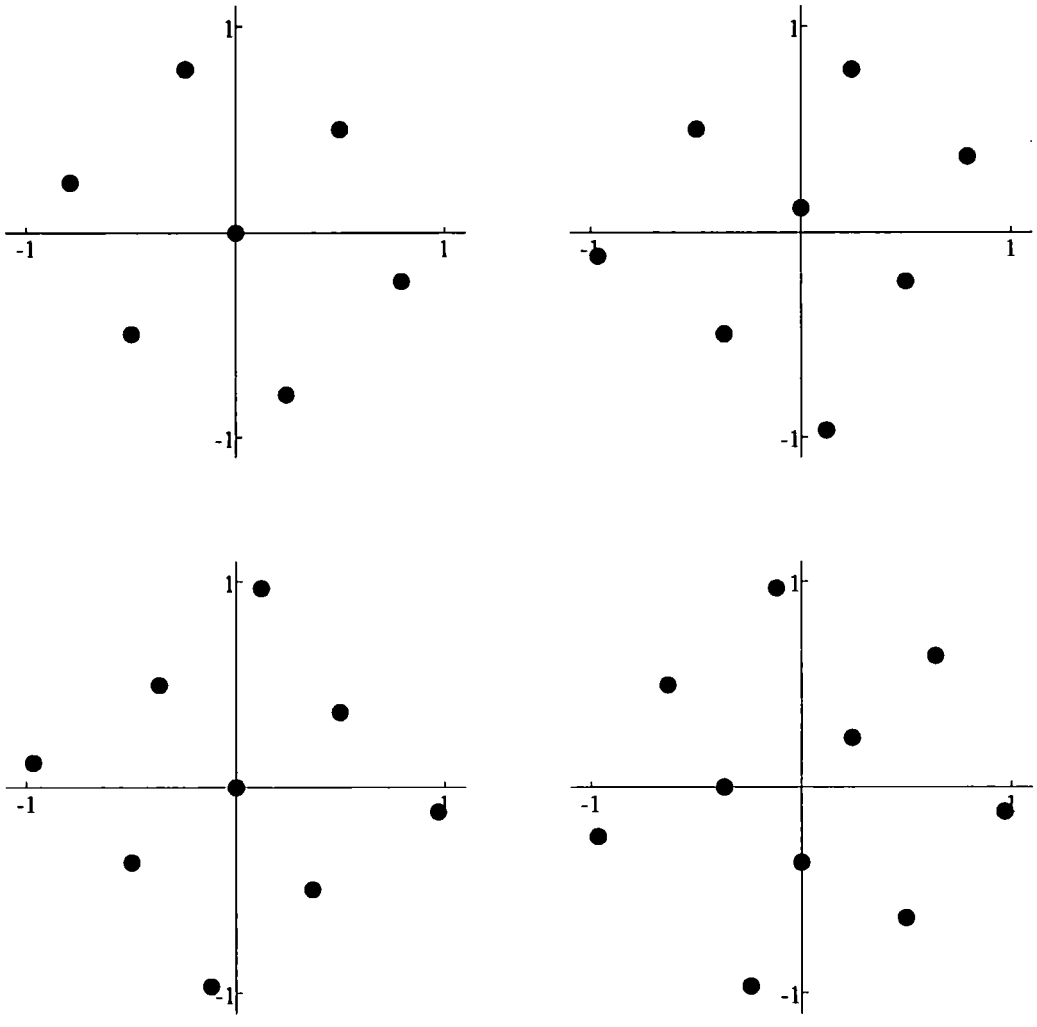

Figure 4.2: 2-dimensional designs on a "normal" grid with seven to ten points

Points	Trace(T)	Design				
1	0.990066	any non-edge point				
2	1.980118	(0.9, 0.9)	(-0.9,-0.9)			
3	2.969742	(-0.9, 0.3)	(0.9, 0.9)	(0.3,-0.9)		
4	3.958110	(-0.9, 0.4)	(-0.4,-0.9)	(0.9,-0.4)	(0.4, 0.9)	
5	4.946558	(-0.9, 0.9)	(-0.9,-0.9)	(0.9,-0.9)	(0.9, 0.9)	(0.0, 0.0)
6	5.931798	(0.9, 0.9)	(0.9,-0.9)	(0.3, 0.3)	(-0.9, 0.9)	(-0.9,-0.9)
		(-0.3,-0.3)				
7	6.915816	(0.9, 0.9)	(0.9,-0.9)	(-0.9, 0.9)	(-0.9,-0.9)	(0.0, 0.5)
		(0.6, 0.0)	(-0.4,-0.3)			
8	7.899789	(0.9, 0.9)	(0.9,-0.9)	(-0.9, 0.9)	(-0.9,-0.9)	(0.6, 0.1)
		(0.1,-0.6)	(-0.6,-0.1)	(-0.1, 0.6)		
9	8.880450	(0.9, 0.9)	(-0.9,-0.9)	(0.9,-0.9)	(-0.4,-0.3)	(0.6, 0.0)
		(0.0, 0.5)	(0.2,-0.7)	(-0.9, 0.2)	(-0.6, 0.9)	
10	9.859666	(-0.9, 0.9)	(0.9, 0.9)	(0.7,-0.9)	(0.4, 0.4)	(-0.2, 0.7)
		(-0.4,-0.9)	(-0.6, 0.1)	(-0.9,-0.6)	(0.9,-0.1)	(0.1,-0.4)
16	15.710535	(-0.4, 0.2)	(0.9,-0.8)	(0.9, 0.6)	(0.2,-0.9)	(-0.9, 0.0)
		(-0.2,-0.7)	(-0.7, 0.5)	(0.3, 0.4)	(-0.9,-0.9)	(-0.9, 0.9)
		(-0.6,-0.4)	(-0.1, 0.8)	(0.6, 0.9)	(0.0,-0.2)	(0.8, 0.0)
		(0.5,-0.5)				

Table 4.3: 2-dimensional designs on a "uniform" grid

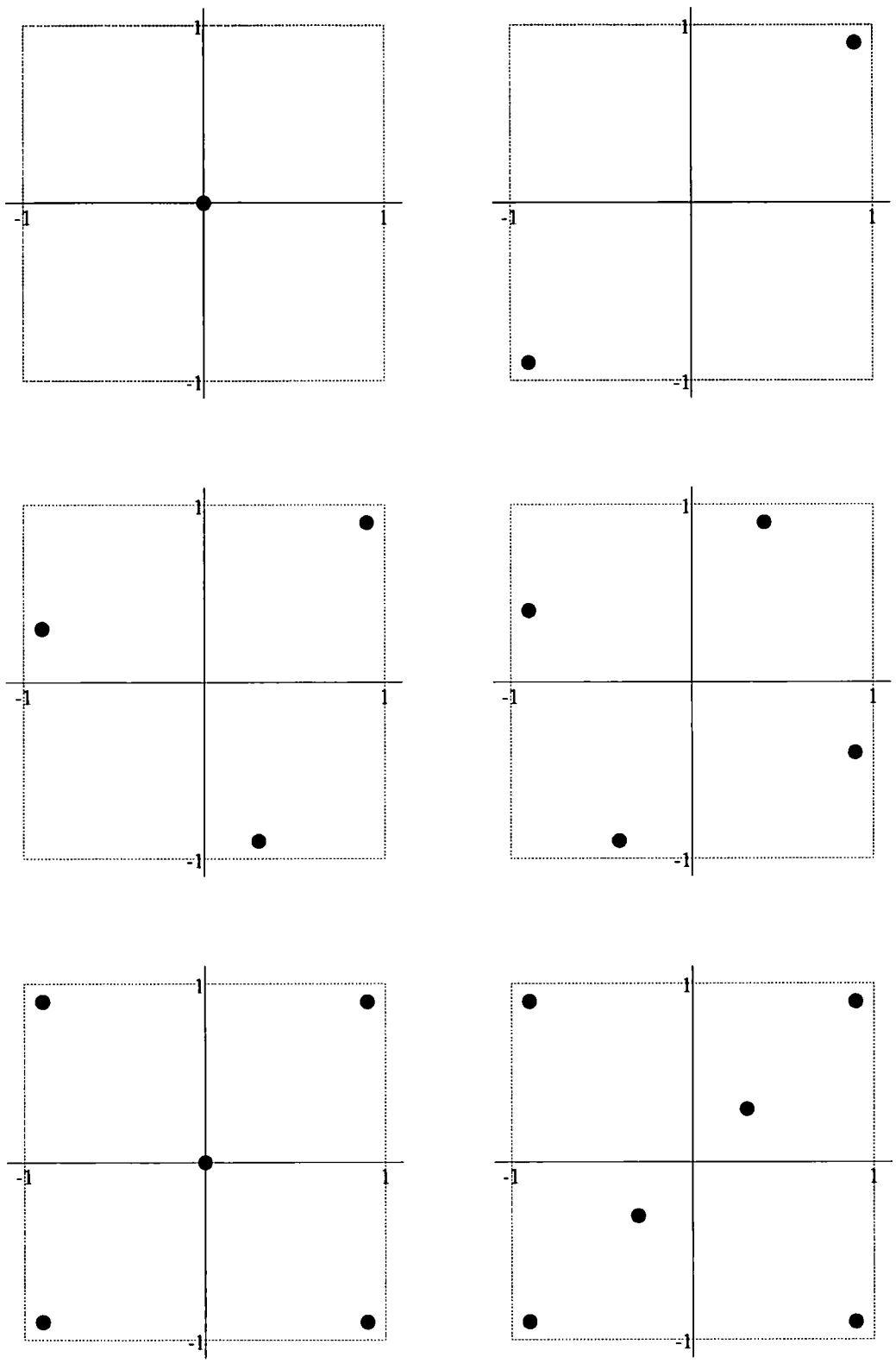

Figure 4.3: 2-dimensional designs on a "uniform" grid with one to six points

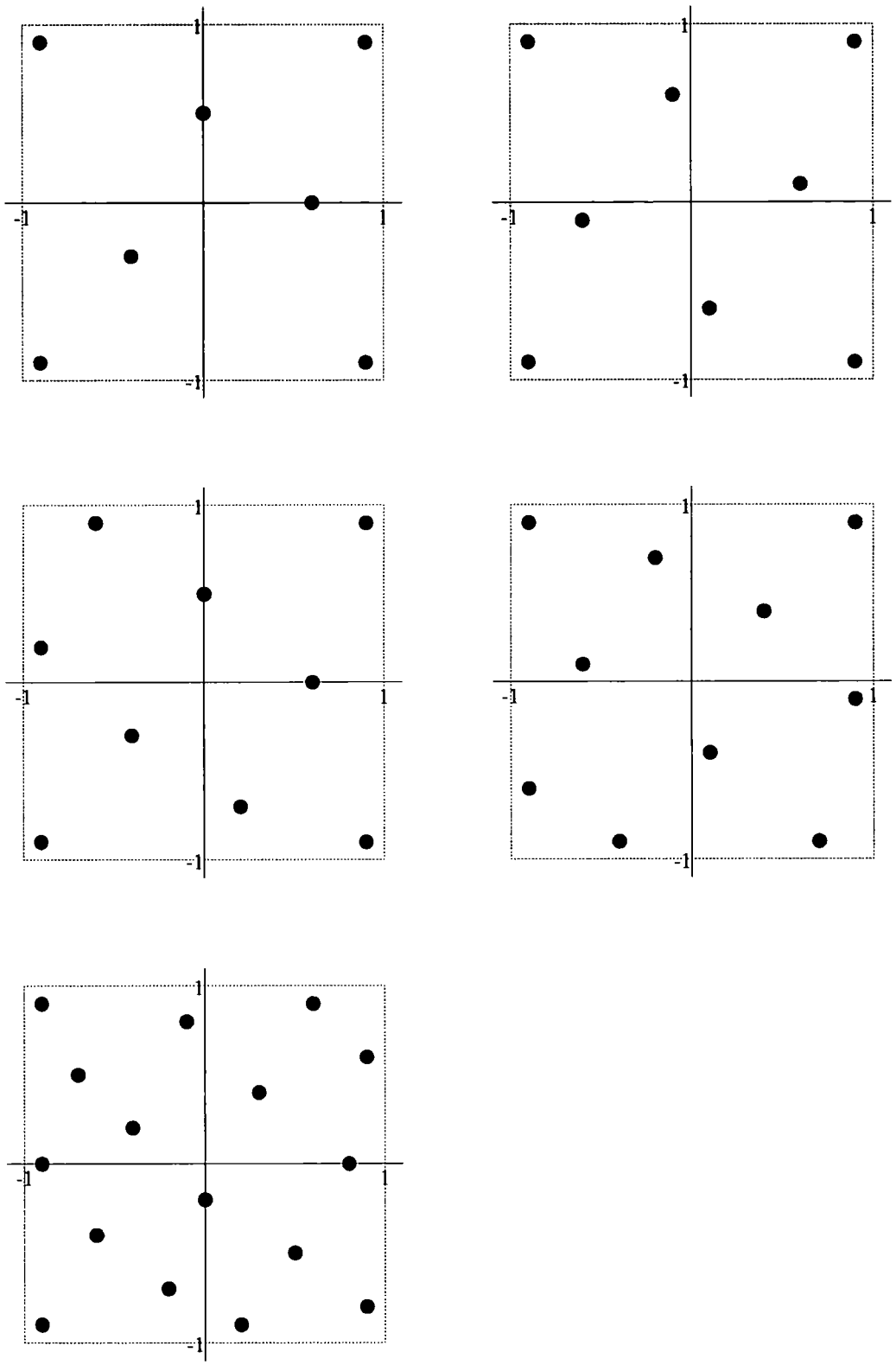

Figure 4.4: 2-dimensional designs on a "uniform" grid with seven to ten and sixteen points

Points	Trace(T)	Design	
1	0.998311	(0.0000, 0.0000, 0.0000)	
2	1.996100	(0.2410, 0.2410, 0.2410)	(-0.2410,-0.2410,-0.2410)
3	2.993659	(0.3661, 0.0000,-0.3661)	(-0.3661,-0.3661, 0.0000)
		(0.0000, 0.3661, 0.3661)	
4	3.991019	(-0.1196,-0.2410, 0.4972)	(-0.4972, 0.1196,-0.2410)
		(0.2410, 0.4972, 0.1196)	(0.3661,-0.3661,-0.3661)
5	4.988282	(0.4972,-0.1196, 0.2410)	(0.0000,-0.4972,-0.1196)
		(-0.2410, 0.2410, 0.4972)	(0.2410, 0.4972,-0.2410)
		(-0.4972, 0.0000,-0.4972)	
6	5.985355	(0.4972, 0.2410, 0.1196)	(-0.2410, 0.6374, 0.0000)
		(0.2410,-0.1196, 0.4972)	(0.0000,-0.2410, 0.6374)
		(-0.1196,-0.4972,-0.2410)	(-0.6374, 0,1196, 0.2410)
7	6.982391	(0.3661, 0.3661,-0.2410)	(-0.6374,-0.2410,-0.1196)
		(-0.2410, 0.6374, 0.1196)	(0.6374,-0.1196, 0.2410)
		(0.1196,-0.6374, 0.0000)	(0.0000, 0.1196, 0.6374)
		(-0.1196, 0.0000,-0.6374)	
8	7.978994	(-0.1196, 0.2410, 0.0000)	(-0.6374,-0.2410,-0.2410)
		(0.3661, 0.6374,-0.2410)	(0.0000,-0.2410, 0.6374)
		(-0.4972, 0.4972, 0.3661)	(0.2410,-0.6374,-0.1196)
		(0.1196, 0.0000,-0.6374)	(0.6374, 0.0000, 0.2410)
9	8.975793	(-0.1196, 0.0000, 0.0000)	(0.3661, 0.4972, 0.2410)
		(-0.4972,-0.3661,-0.3661)	(0.0000,-0.2410, 0.6374)
		(0.6374,-0.1196,-0.2410)	(0.1196, 0.2410,-0.6374)
		(-0.2410, 0.7916,-0.2410)	(0.2410,-0.6374, 0.1196)
		(-0.6374, 0.2410, 0.3661)	
10	9.972339	(0.1196,-0.2410,-0.1196)	(-0.1196, 0.6374, 0.1196)
		(0.3661, 0.3661,-0.3661)	(0.3661,-0.4972, 0.3661)
		(-0.7916, 0.2410,-0.1196)	(-0.1196, 0.0000,-0.7916)
		(0.1196, 0.2410, 0.7916)	(0.7916, 0.0000, 0.1196)
		(-0.3661,-0.6374,-0.3661)	(-0.4972,-0.1196, 0.4972)

Table 4.4: 3-dimensional designs on a "normal" grid

Points	Trace(T)	Design		
1	0.999010	any non-edge point		
2	1.998020	(-0.9,-0.9,-0.9)	(0.9, 0.9, 0.9)	
3	2.997025	(-0.9, 0.9,-0.9)	(0.9, 0.9, 0.9)	(0.9,-0.9,-0.9)
4	3.996031	(0.9,-0.9,-0.9)	(0.9, 0.9, 0.9)	(-0.9,-0.9, 0.9)
		(-0.9, 0.9,-0.9)		
5	4.994998	(0.9,-0.9,-0.2)	(-0.9, 0.9,-0.2)	(-0.9,-0.9, 0.9)
		(0.0, 0.0,-0.9)	(0.9, 0.9, 0.9)	
6	5.993964	(-0.9,-0.9, 0.9)	(0.9,-0.9, 0.1)	(-0.4,-0.5,-0.9)
		(0.9, 0.9,-0.9)	(0.4, 0.5, 0.9)	(-0.9, 0.9,-0.1)
7	6.992880	(0.9, 0.9,-0.9)	(-0.9,-0.8,-0.9)	(0.9,-0.9,-0.1)
		(-0.9, 0.9, 0.4)	(-0.1, 0.1,-0.4)	(-0.5,-0.9, 0.9)
		(0.8, 0.5, 0.9)		
8	7.991753	(0.9,-0.3,-0.9)	(-0.9, 0.8,-0.9)	(0.9,-0.9, 0.9)
		(-0.9,-0.7, 0.7)	(-0.5, 0.9, 0.9)	(0.9, 0.9,-0.1)
		(0.1, 0.0, 0.3)	(-0.4,-0.9,-0.7)	
9	8.990604	(-0.9, 0.9, 0.3)	(0.9, 0.9, 0.9)	(0.9,-0.9, 0.4)
		(-0.9,-0.9, 0.9)	(0.3, 0.9,-0.7)	(0.9,-0.3,-0.9)
		(0.0, 0.0, 0.7)	(-0.9, 0.1,-0.9)	(-0.3,-0.9,-0.5)
10	9.989434	(-0.1,-0.9, 0.9)	(-0.9, 0.9, 0.9)	(-0.9,-0.9,-0.8)
		(0.9, 0.9,-0.6)	(0.9, 0.2, 0.9)	(-0.7, 0.9,-0.9)
		(0.4,-0.2,-0.9)	(-0.8,-0.3, 0.3)	(0.0, 0.6, 0.1)
		(0.9,-0.9,-0.1)		

Table 4.5: 3-dimensional designs on a "uniform" grid

Chapter 5

Looking For Nothing

As mentioned in Chapter 1, one of the more common problems when we are considering computer experiments modelling complicated processes is that of matching the output of the model to the real observed data, by adjusting the model parameters, for example the flame dynamics model of Sacks *et al*[1989a]. In the papers mentioned in Chapter 2 that cover this topic of calibration, the methods used all consist of finding an approximation to the function, and then finding points where this approximation matches the observed data.

Here, we would like to explicitly build into our approximation information about the model that is relevant to finding a solution to this problem, which we can express as

$$\boldsymbol{y}(\boldsymbol{x}) = \boldsymbol{y}_0 \tag{5.1}$$

where $\boldsymbol{y}_0$ is the observed data. We can w.l.o.g. replace this with the simpler problem of solving $\boldsymbol{y}(\boldsymbol{x}) = \mathbf{0}$, by rewriting $\boldsymbol{y}(\boldsymbol{x})$ as $\boldsymbol{y}(\boldsymbol{x}) - \boldsymbol{y}_0$. We would like to include prior knowledge about the location of the zero, that is our prior prevision (expectation), and a measure of uncertainty (variance). Other information could be included, for example the slope of the function near the zero.

In the next few sections, we will consider the simplest case of this problem where

the response function $y(\cdot)$ and the parameter x are both univariate. This is a typical problem considered by numerical analysts, and so there is a plethora of numerical methods already in existence; for example

Bisection: for a function on an interval $[a, b]$, whose values at the two end points are of opposite sign. To home in on the zero, we repeatedly halve the interval, keeping the half in which the function values at the end points have opposite sign, and hence contains the zero.

Secant: for functions of the same type as for the bisection method. The interval is split at the point where the straight line through the function evaluations at the two end points cuts the axis, and again we keep the half in which the function values at the end points have opposite sign.

Newton Raphson: for differentiable functions. Given a starting value x_0, we replace it by the point where the tangent line to the function at x_0 crosses the axis, and repeat this until we converge on a solution.

Inverse Interpolation: for any function. The function is approximated by some means, and then we solve (the easier) problem of finding the roots of the approximant.

5.1 Modelling the problem - the univariate case

To further simplify this problem, we will make an additional initial assumption that the function has just one zero in the region of interest. Some of the methods to be produced will identify multiple zeros (even under the assumption that there is just one). A much harder problem of estimating the number of zeros the interval is omitted, if this was possible, we could build this information into the model.

We first consider the construction of the model of the function. As we are considering a function with a zero, we choose to explicitly include this in the model. If we let X_0 denote the location of the zero, the model must have the following property

$$Y(X_0) = 0 \tag{5.2}$$

As most functions we will consider are continuous then we should assume that our model of the response is also continuous.

The first property (5.2) can be modelled in two ways;

i) We can assume $Y(x)$ has $(x - X_0)$ as a divisor i.e.

$$Y(x) = B(x)(x - X_0) \tag{5.3}$$

where $B(x)$ is a continuous function, see figure 5.1 $B(X_0)$ represents the slope of the function at its zero.

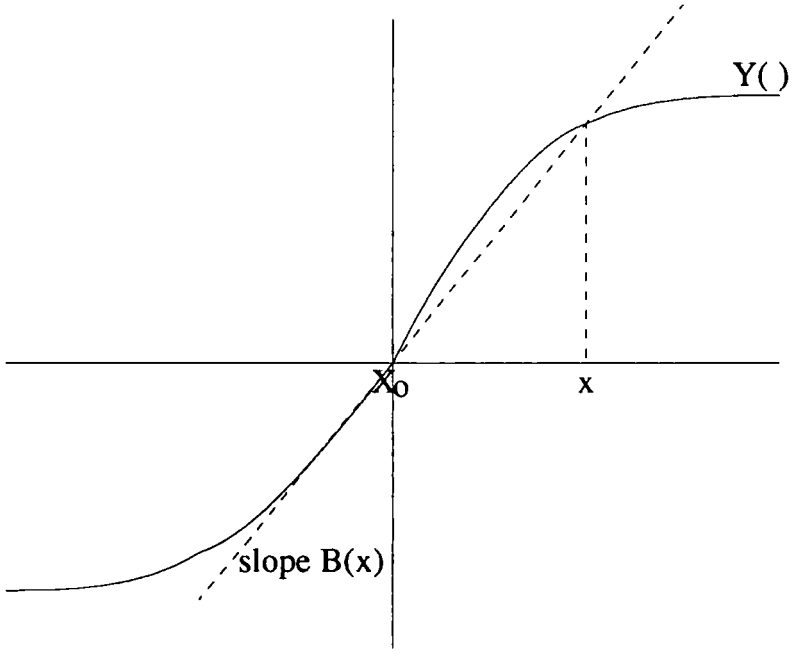

Figure 5.1: $x - X_0$ factor model for $Y(x)$

ii) In general we can write $Y(x)$ in the following form

$$Y(x) = G(x) - G(X_0) \tag{5.4}$$

where $G(x)$ is again continuous, and so when we model $G(x)$ where we do not need to put additional constraints on $G(X_0)$. We will look at this model again

in section 5.6.

The continuity assumption can be modelled by assuming that the prevision of $Y(x)$ as a function of x is continuous, and that the correlation between two values of $Y(x)$ is also continuous. We will expand on this in the next section.

5.2 The univariate model

In this and the next few sections, we will assume the model is as in equation (5.3), and model the "slope" process $B(\cdot)$. As we have already stated $B(x)$ should be a continuous function of x. For this to happen its prevision should be continuous, and that the correlation between the "slope" $B(\cdot)$ at any two points x and x_* should also be a continuous function of x and x_*.

We assume for simplicity that the "slope" process $B(\cdot)$ is stationary, so that the prevision of $B(x)$ does not depend on x and therefore we have

$$\mathrm{P}(B(x)) = b, \tag{5.5}$$

for all x in the region of interest. Our prior specification for b is based on our prior knowledge about the slope at the zero.

We assume that the correlation of the "slope" $B(\cdot)$ is a continuous monotonically decreasing function of the distance between two points. If we believe $y(\cdot)$ to be a continuously differentiable function we would also like the correlation function to be continuously differentiable. In what follows we will use $\rho(d) = e^{-\theta d^2}$ ($\rho_2(\cdot)$ from Chapter 2) as the correlation function, where d is the distance between the two points. This condition is not essential, and we will look at some other correlation functions later for which this does not hold. We base our prior belief about the variance of $B(x)$, σ_b^2, on the variation of the slope at the "zero".

Finally we specify our beliefs about X_0, its location $\mathrm{P}(X_0) = \mu_0$ and uncertainty

$\mathrm{Var}(X_0) = \sigma_0^2$. These assumptions and prior beliefs imply beliefs about the $Y(\cdot)$s,

$$\begin{aligned}
\mathrm{P}(Y(x)) &= b(x-\mu_0) && (5.6)\\
\mathrm{Cov}(Y(x),Y(x_*)) &= \left(\sigma_0^2+(x-\mu_0)(x_*-\mu_0)\right)\sigma_b^2\rho(x-x_*)+b^2\sigma_0^2 && (5.7)\\
\mathrm{Cov}(X_0,Y(x)) &= -b\sigma_0^2 && (5.8)\\
\mathrm{Cov}(Y(x),B(x_*)) &= \sigma_b^2(x-\mu_0)\rho(x-x_*) && (5.9)
\end{aligned}$$

These seem natural, the variance of $Y(\cdot)$ grows quadratically as we move away from the location of the zero, which is what we would expect if we were considering a linear model; the sign in (5.8), highlights that if we observe a value higher than expected then (with a positive slope) the zero is more likely to be to the left of the original guess; the correlation in the function evaluations tends to 0 as the distance between them increases.

We can look at plots of the correlation (figures 5.2 and 5.3) and covariance (figure 5.4) of the $Y(\cdot)$s for various values of b, and θ, under this model (using $\rho_2(\cdot)$), and $\mu_0 = 0$, $\sigma_0^2 = 1$ and $\sigma_b^2 = 1$ (we can use changes of scale and location to transform the variables so that μ_0, σ_0^2 and σ_b^2 are replaced by 0, 1 and 1 respectively). These give us further insight into the structure we have implied.

We can also produce sample realizations of the random processes with these first and second order structures (We extend our prior beliefs about the $B(\cdot)$s to a full prior distribution, by assuming they are multivariate normal), to give us some idea of functions which match our model precisely.

If we let $Y_1,\ldots,Y_n$ be the unknown function values at $x_1,\ldots,x_n$ we can adjust our beliefs for each of the quantities of interest X_0, $B(x)$ and $Y(x)$. We will first define some notations to be used throughout the rest of this chapter.

$$\begin{aligned}
\boldsymbol{x} &= (x_1,\ldots,x_n)^{\mathrm{T}}\\
\boldsymbol{Y} &= (Y_1,\ldots,Y_n)^{\mathrm{T}}
\end{aligned}$$

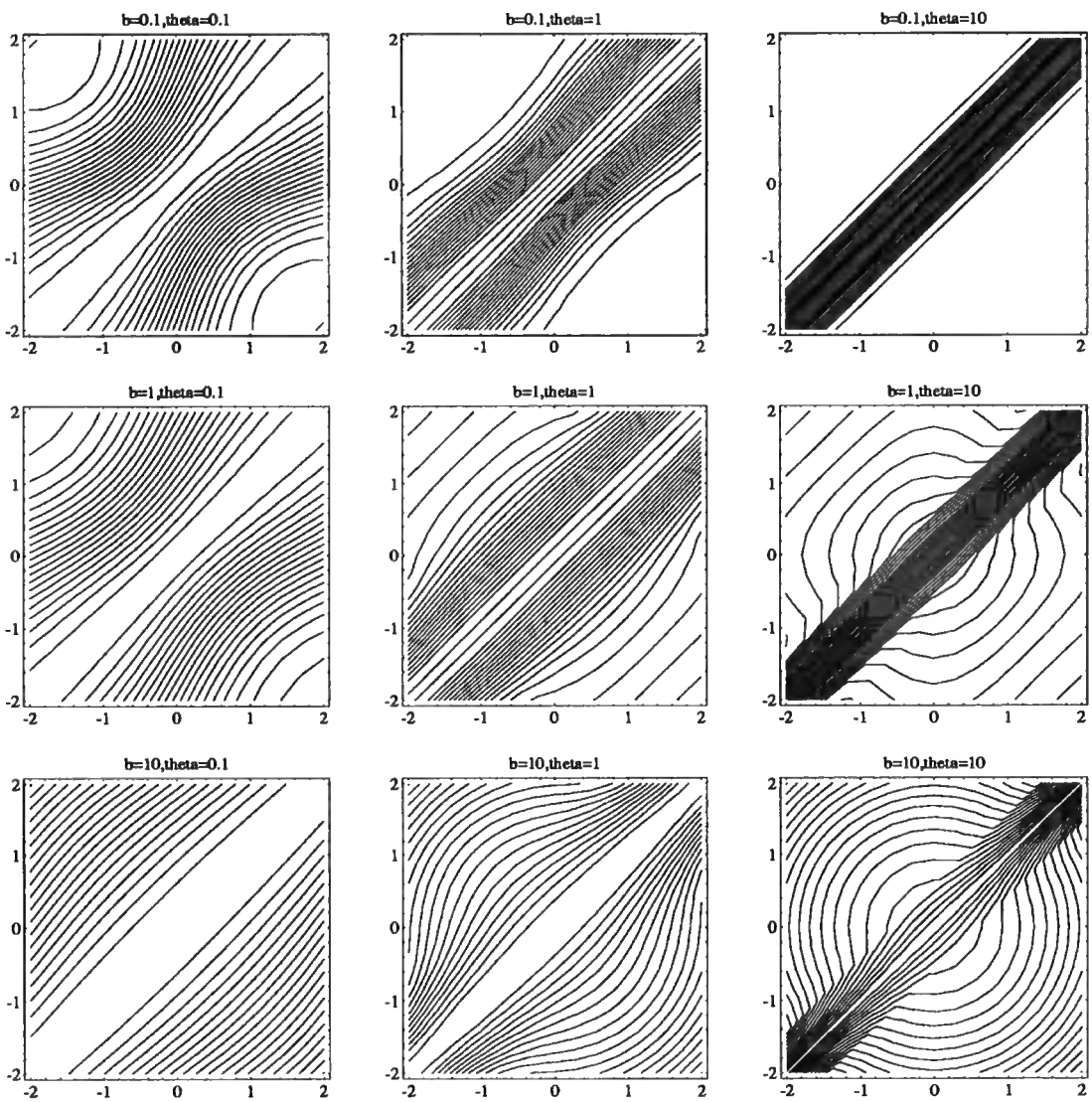

Figure 5.2: Correlation between $Y(x)$ and $Y(x^*)$, using $\rho_2(d)$ as the slope correlation

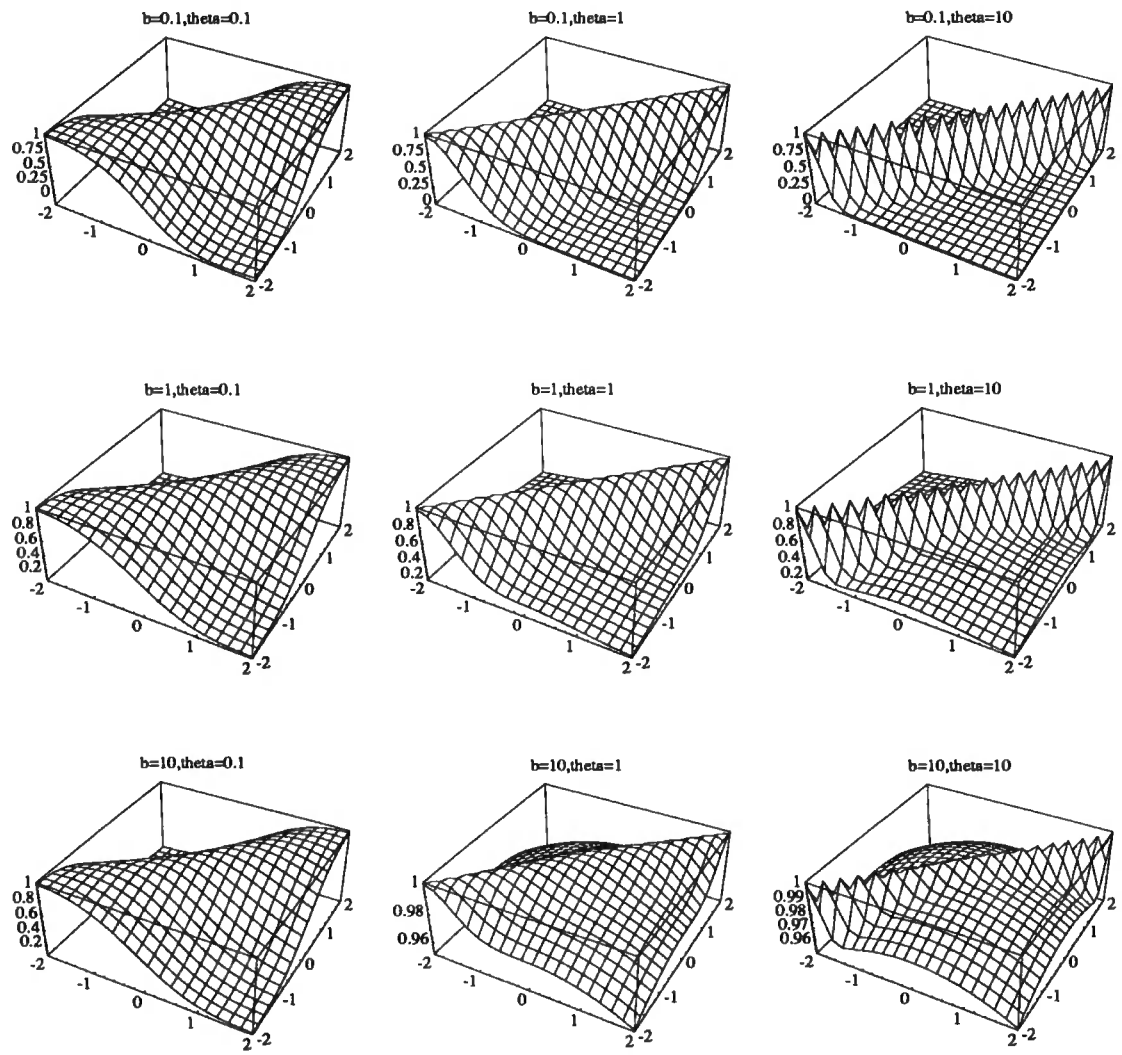

Figure 5.3: Correlation between $Y(x)$ and $Y(x^*)$, using $\rho_2(d)$ as the slope correlation

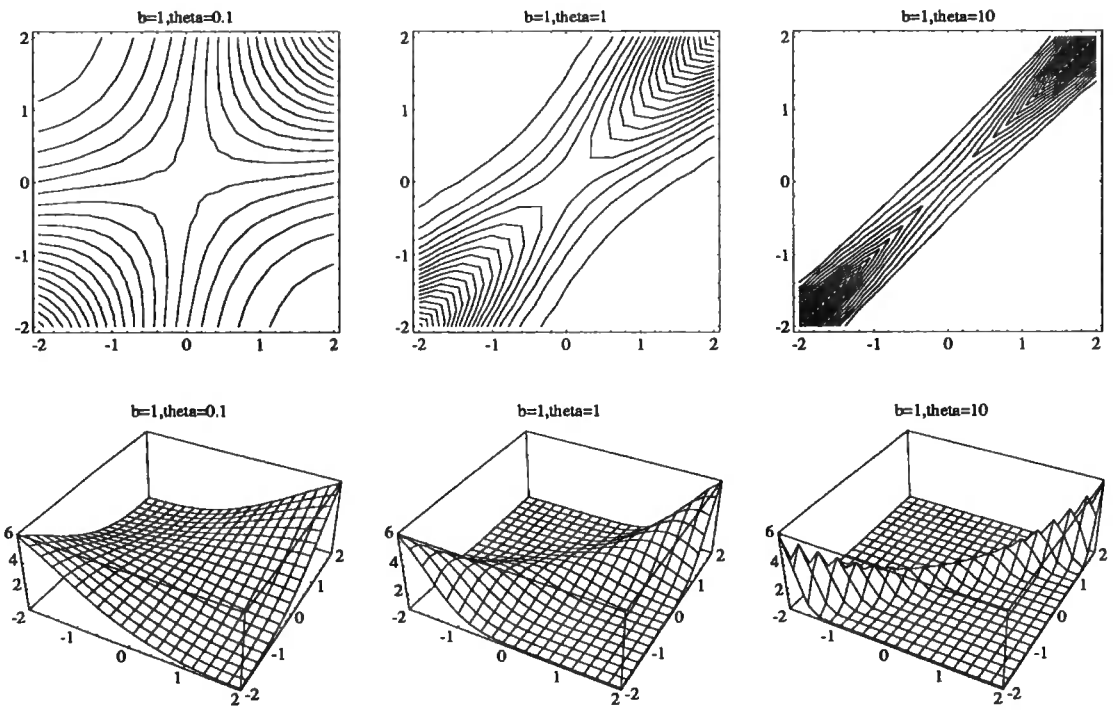

Figure 5.4: Covariance between $Y(x)$ and $Y(x^*)$, using $\rho_2(d)$ as the slope correlation

$$
\begin{aligned}
c(u,v) &= \rho(u-v)((u-\mu_0)(v-\mu_0)+\sigma_0^2) \\
C_{ij} &= c(x_i,x_j) \\
\boldsymbol{c} &= (c(x_*,x_1),\ldots,c(x_*,x_n))^{\mathrm{T}} \\
\tilde{\boldsymbol{c}} &= (c(\tilde{x}_*,x_1),\ldots,c(\tilde{x}_*,x_n))^{\mathrm{T}} \\
\boldsymbol{d} &= ((x_1-\mu_0)\rho(x_*,x_1),\ldots,(x_n-\mu_0)\rho(x_*-x_n))^{\mathrm{T}} \\
\tilde{\boldsymbol{d}} &= ((x_1-\mu_0)\rho(\tilde{x}_*,x_1),\ldots,(x_n-\mu_0)\rho(\tilde{x}_*-x_n))^{\mathrm{T}}
\end{aligned}
$$

(Note $\sigma_b^2 c(x,x_*)$ is the covariance between $Y(x)$ and $Y(x_*)$ if b=0)

The Bayes linear estimates for X_0, $B(\cdot)$ and $Y(\cdot)$ given our prior beliefs can be written as

$$\mathrm{P}_{\boldsymbol{Y}}(X_0) = \frac{\sigma_b^2\mu_0 - \sigma_0^2\mathbf{1}^{\mathrm{T}}C^{-1}(\boldsymbol{Y}-b\boldsymbol{x})b}{\sigma_b^2+\sigma_0^2\mathbf{1}^{\mathrm{T}}C^{-1}\mathbf{1}b^2} \tag{5.10}$$

$$\mathrm{P}_{\boldsymbol{Y}}(B(x_*)) = b+\boldsymbol{d}^{\mathrm{T}}C^{-1}(\boldsymbol{Y}-b[\boldsymbol{x}-P_{\boldsymbol{Y}}(X_0)\mathbf{1}]) \tag{5.11}$$

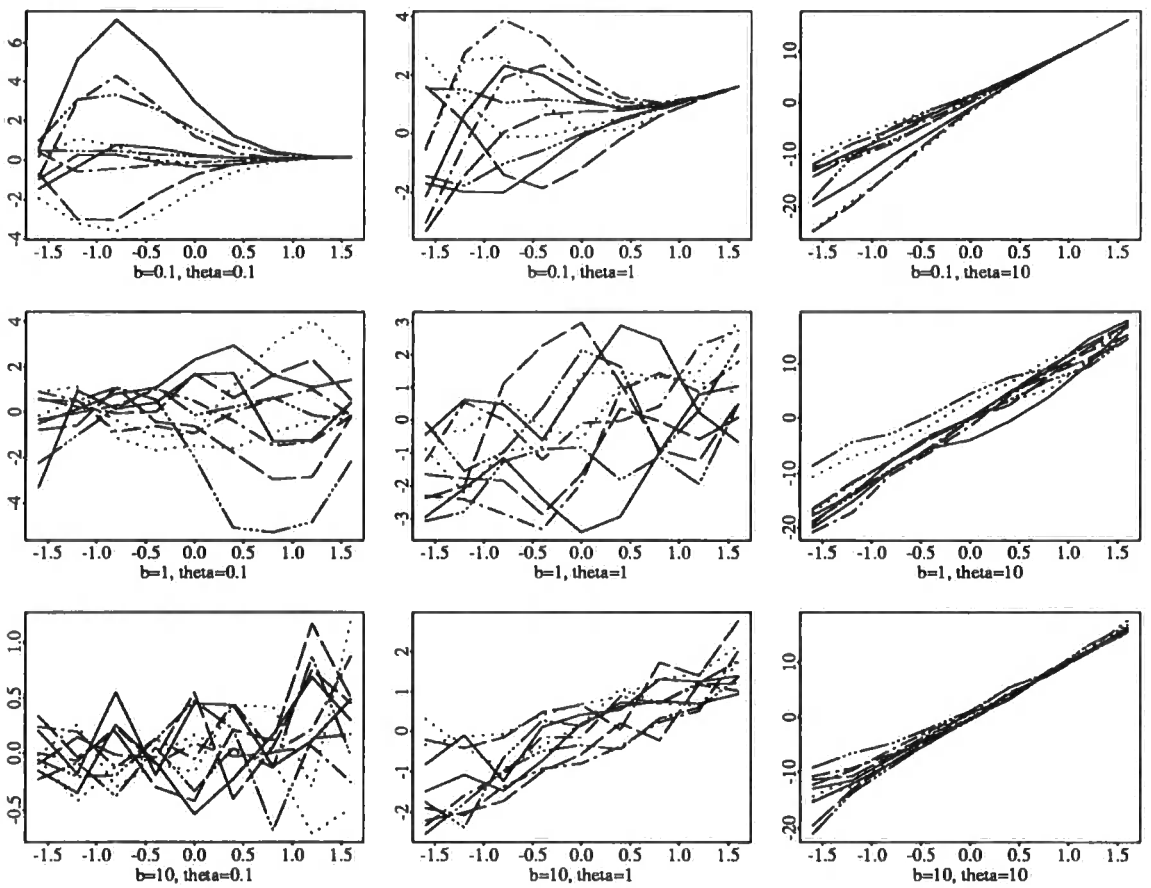

Figure 5.5: Traces of $Y(x)$, using $\rho_2(d)$ as the slope correlation

$$\mathrm{P}_{\boldsymbol{Y}}(Y(x_*)) = b[x_* - \mathrm{P}_{\boldsymbol{Y}}(X_0)] + \boldsymbol{c}^{\mathrm{T}}C^{-1}(\boldsymbol{Y} - b[\boldsymbol{x} - \mathrm{P}_{\boldsymbol{Y}}(X_0)\mathbf{1}]) \quad (5.12)$$

and their adjusted variances and covariances are

$$\mathrm{Var}[X_0/\boldsymbol{Y}] = \frac{\sigma_0^2\sigma_b^2}{\sigma_b^2 + \sigma_0^2\mathbf{1}^{\mathrm{T}}C^{-1}\mathbf{1}b^2} \quad (5.13)$$

$$\mathrm{Cov}([X_0/\boldsymbol{Y}],[Y(x_*)/\boldsymbol{Y}]) = -b(1 - \mathbf{1}^{\mathrm{T}}C^{-1}\boldsymbol{c})\mathrm{Var}[X_0/\boldsymbol{Y}] \quad (5.14)$$

$$\mathrm{Cov}([B(x_*)/\boldsymbol{Y}],[B(\tilde{x}_*)/\boldsymbol{Y}]) = \sigma_b^2\left(\rho(x_* - \tilde{x}_*) - \boldsymbol{d}^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{d}}\right) + \boldsymbol{d}^{\mathrm{T}}C^{-1}\mathbf{1}\mathbf{1}^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{d}}\mathrm{Var}[X_0/\boldsymbol{Y}]b^2 \quad (5.15)$$

$$\mathrm{Cov}([Y(x_*)/\boldsymbol{Y}],[Y(\tilde{x}_*)/\boldsymbol{Y}]) = \sigma_b^2\left((\sigma_0^2 + (x_*-\mu_0)(\tilde{x}_*-\mu_0))\rho(x_*-\tilde{x}_*) - \boldsymbol{c}^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{c}}\right) + (1-\mathbf{1}^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{c}})(1-\boldsymbol{c}^{\mathrm{T}}C^{-1}\mathbf{1})\mathrm{Var}[X_0/\boldsymbol{Y}]b^2 \quad (5.16)$$

$$\mathrm{Cov}([B(x_*)/\boldsymbol{Y}],[Y(\tilde{x}_*)/\boldsymbol{Y}]) = \sigma_b^2\left((\tilde{x}-\mu_0)\rho(x_*-\tilde{x}_*) - \boldsymbol{d}^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{c}}\right) \boldsymbol{d}^{\mathrm{T}}C^{-1}\mathbf{1}(\mathbf{1}^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{c}}-1)\mathrm{Var}[X_0/\boldsymbol{Y}]b^2 \quad (5.17)$$

We can now make some observations about these results;

The posterior estimate for X_0 is a weighted average of the prior estimate and an estimate based on the data (of the form.

The posterior estimates for both $B(x)$ and $Y(x)$ both include $(x - P_{\boldsymbol{Y}}(X_0))$ in them, as opposed to $x - \mu_0$ in the prior estimate, and the covariances include $\mathrm{Var}[X_0/\boldsymbol{Y}]$.

The adjusted covariances between the $B(\cdot)$s and $Y(\cdot)$ are all linear combinations of σ_b^2 and $\mathrm{Var}[X_0/\boldsymbol{Y}]b^2$.

$(C^{-1}\tilde{\boldsymbol{c}})_j = \delta_{ij}$ at the design point x_i.

Before we go on to look at the question of design, we will examine the predictors for fixed designs. We will use designs with one, three, five and nine points equally spaced in the interval $[-1,1]$, with $\sigma_0 = \sigma_b = b = \theta = 1$, and $\mu_0 = 0$. As a test function we use $\sin(x/2 - 0.1)$, which has a zero at $x = 0.2$. With these designs, we obtain the adjusted previsions and variances for X_0 listed in Table 5.1. We can also examine our adjusted previsions and covariances for $B(\cdot)$ and $Y(\cdot)$, this is most easily

performed graphically, see Figures 5.6 (prevision) and 5.7 (covariance). Examining these we see that: although the covariances of the $Y(\cdot)$ is reduces by a factor of about 90% for every additional function evaluation we make, the covariances of the $B(\cdot)$ only decrease by at most 20% for each evaluation – the same is true for the posterior variance of X_0; the correlation between $Y(x)$ either side of a design point is negative, which is as we would expect; and although the covariance between two $Y(\cdot)$s drops to zero near the design points this is not the case for the covariance of the $B(\cdot)$s.

Design	$P_{\mathbf{Y}}(X_0)$	$\mathrm{Var}[X_0/\mathbf{Y}]$
$\{0\}$	0.0499167	0.5
$\{-1, 0, 1\}$	0.0556568	0.406155
$\{-1, -\frac{1}{2}, 0, \frac{1}{2}, 1\}$	0.555152	0.405009
$\{-1, -\frac{3}{4}, -\frac{1}{2}, -\frac{1}{4}, 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\}$	0.0552358	0.392670

Table 5.1: Posterior mean and variance of X_0 given function evaluations

5.3 The design of the experiment

Our model is a basis for predicting $y(\cdot)$ at any value of x, using the posterior prevision of $Y(\cdot)$ as our estimate of it. The design problem now is how to choose the values of x at which to evaluate $Y(\cdot)$ so that we can estimate X_0 as efficiently as possible. If the function we are trying to approximate is cheap then we have no real limits on how many evaluations we can make, so the design question is not as important as when the function is expensive.

We have considered three main design approaches:

a completely pre-determined "optimal" design, choosing all the points before we make any function evaluations;

a sequential one-step-ahead approach;

and a mixture of the two, constituting an initial "optimal" pre-determined block design followed by a sequential refinement (in which new sets of points are added to the design, not necessarily one at a time.)

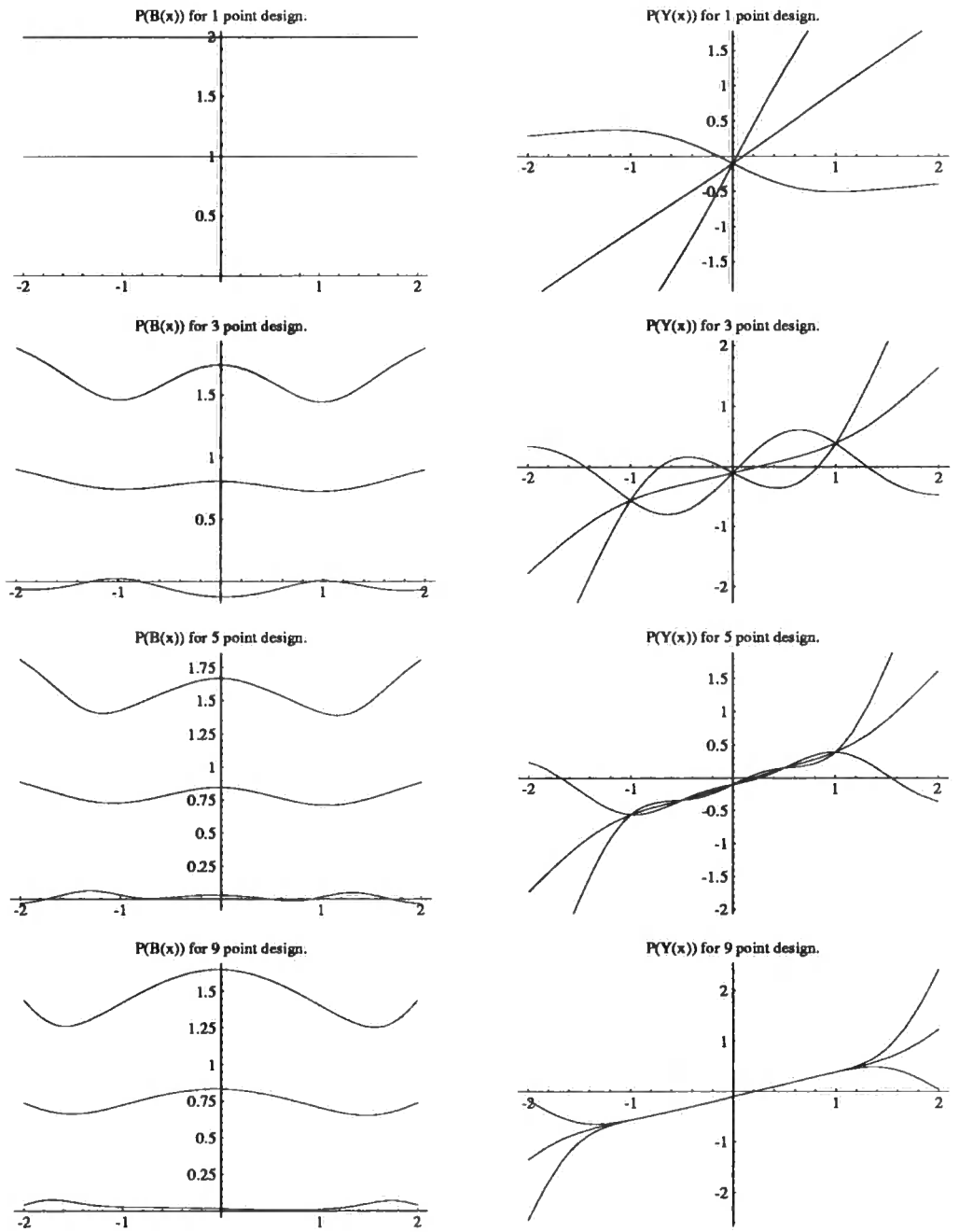

Figure 5.6: Adjusted previsions of $B(\cdot)$ and $Y(\cdot)$, with one standard deviation error lines

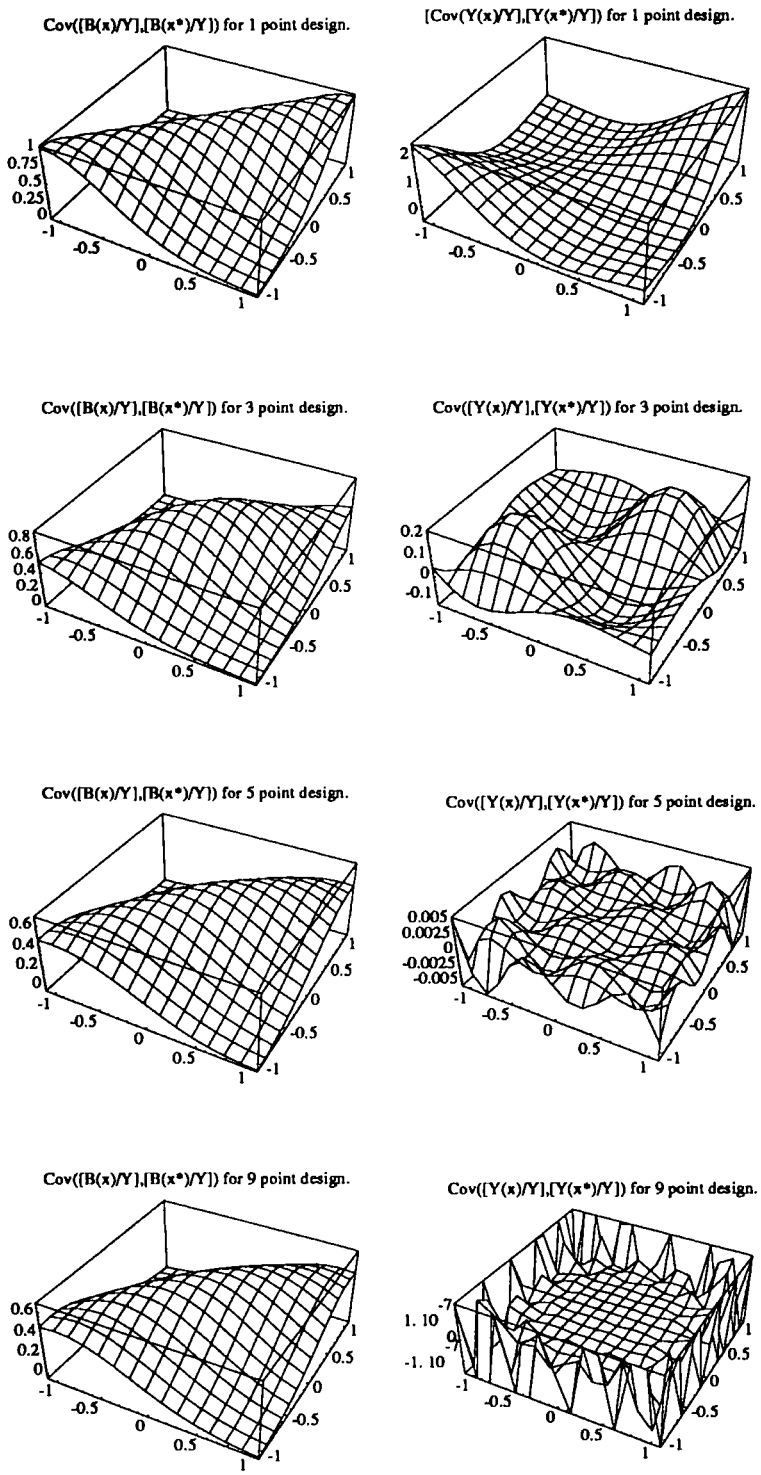

Figure 5.7: Adjusted covariances of $B(\cdot)$ and $Y(\cdot)$

The pre-determined designs have the advantage that most of the difficult calculations can be computed in advance to a very high degree of accuracy, especially the inverse of the correlation matrix. However in some situations these fixed designs are too rigid and do not take into account any extra information that becomes available every time the function is evaluated. For cheap functions though these designs can be useful, because any complicated arithmetic that we might have to compute to get the "best" estimator for a set of function evaluations must take less computer time and power than a simpler approach which might take more function evaluations. Another downside to these designs is the large amount of computing power required to compute them initially, with the sequential designs all the optimization required has the same dimensionality as $\boldsymbol{x}$ (in this case 1), but with fixed designs it is multiplied by the number of design points and soon the problem of finding these "optimal" designs becomes more difficult to solve than the original problem. If we have a fixed design we can consider the sub-problem of choosing the order of the design points, as we need not randomize the ordering to remove "time dependent" biases as none can appear.

"One-step-ahead" sequential designs have the obvious advantages that, after each function evaluation, our beliefs about the function are modified, and so we modify the design to take this information into account. Again with cheap functions we need to keep sequential designs simple to be as efficient as more naive methods that require more function evaluations. On the other hand, when the function we are approximating is expensive, we can in general spend more time choosing the next design point, for if we can reduce the total number of function evaluations great savings of time and money may be possible.

We can construct a compromise between these two design extremes, by having and initial block followed by either a sequence of additional design points, or additional points. The first of these combine the advantages of the two previous design types, the ability to compute part of the inverse of the variance matrix before, and then the simplicity of the "one-step-ahead" sequential process. On the other hand "block-sequential" designs are not as good, as they are generally more complicated

to compute, and also suffer from the problem of making "redundant" function evaluations.

We are mainly concerned with the expensive function, so we will only consider the sequential approach. For this we need a design criterion or criteria to decide which is the best point to choose next. Good criteria should:

1) Use as much information from the data received to choose the next point

2) Quickly home in on the zero.

3) Initially explore a sufficiently large area of the design space to make sure it does not miss the zero.

5.3.1 Test functions

To compare various methods we shall include some simple test problems, obviously these are cheap functions, but they can be treated as expensive ones. They are as follows

$$\begin{aligned} f_1(x) &= \sin\left(\frac{x-0.2}{2}\right) \\ f_2(x) &= (x-0.2)^{11} + (0.7)^{11}. \end{aligned}$$

$f_1(\cdot)$ is a simple near straight line with a zero at 0.2, whereas $f_2(\cdot)$ is a more difficult function for many of zero solvers as the gradient changes greatly throughout the interval, being approximately 0 over a large part of the design – for example Newton-Raphson starting at $x = 0$ takes nearly one hundred function evaluations to loacate the zero. It has a zero at -0.5. Figure 5.8 shows graphs of these two functions.

5.3.2 The naive estimate

Using a sequential approach we can re-write our equation for $\mathrm{P}_{\boldsymbol{Y}Y_*}(X_0)$ after $n+1$ function evaluations $Y_1, \ldots, Y_n, Y_*$ at $x_1, \ldots, x_n, x_*$, in terms of our beliefs after n

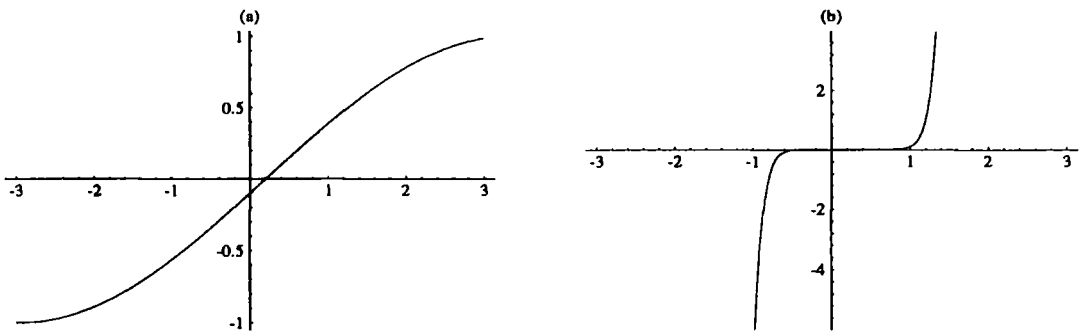

Figure 5.8: (a) - $f_1(x) = \sin\left(\frac{x-0.2}{2}\right)$, (b) - $f_2(x) = (x+0.5)^{11} - (0.2)^{11}$

observations,

$$\mathrm{P}_{\boldsymbol{Y},Y_*}(X_0) = \frac{\begin{array}{r}[\sigma_b^2(\sigma_0^2 + (x_* - \mu_0)^2) - \boldsymbol{c}^{\mathrm{T}} C_n^{-1}\boldsymbol{c}]\mathrm{P}_{\boldsymbol{Y}}(X_0) - \\ b(1 - \mathbf{1}^{\mathrm{T}} C_n^{-1}\boldsymbol{c})\mathrm{Var}[X_0/\boldsymbol{Y}](Y_* - bx_* - \boldsymbol{c}^{\mathrm{T}} C_n^{-1}(\boldsymbol{Y} - b\boldsymbol{x}))\end{array}}{\sigma_b^2(\sigma_0^2 + (x_* - \mu_0)^2) - \boldsymbol{c}^{\mathrm{T}} C_n^{-1}\boldsymbol{c} + b^2(1 - \mathbf{1}^{\mathrm{T}} C_n^{-1}\boldsymbol{c})^2\mathrm{Var}[X_0/\boldsymbol{Y}]} \tag{5.18}$$

Therefore we no longer have to invert the variance matrix when adding the next design point if we already have the inverse for n points. We can sequentially calculate the inverse as well,

$$C_{n+1}^{-1} = \begin{pmatrix} C_n^{-1} + \frac{C_n^{-1}\boldsymbol{c}\boldsymbol{c}^{\mathrm{T}} C_n^{-1}}{c(x_*,x_*) - \boldsymbol{c}^{\mathrm{T}} C_n^{-1}\boldsymbol{c}} & -\frac{C_n^{-1}\boldsymbol{c}}{c(x_*,x_*) - \boldsymbol{c}^{\mathrm{T}} C_n^{-1}\boldsymbol{c}} \\ -\frac{\boldsymbol{c}^{\mathrm{T}} C_n^{-1}}{c(x_*,x_*) - \boldsymbol{c}^{\mathrm{T}} C_n^{-1}\boldsymbol{c}} & \frac{1}{c(x_*,x_*) - \boldsymbol{c}^{\mathrm{T}} C_n^{-1}\boldsymbol{c}} \end{pmatrix}$$

A criterion to compliment this is to choose the design minimizing the adjusted variance of X_0, which from Section 5.2 is

$$\mathrm{Var}[X_0/Y] = \frac{\sigma_0^2\sigma_b^2}{\sigma_b^2 + \sigma_0^2 \mathbf{1}^{\mathrm{T}} C^{-1}\mathbf{1}}$$

which is the same as maximizing $\mathbf{1}^{\mathrm{T}} C^{-1}\mathbf{1}$. In the sequential design problem, given

function evaluations at n previous points, we can choose the next point x_* to maximize

$$\begin{aligned} \mathbf{1}^{\mathrm{T}} C_{n+1}^{-1} \mathbf{1} &= \begin{pmatrix} \mathbf{1}^{\mathrm{T}} & 1 \end{pmatrix} \begin{pmatrix} C_n & \boldsymbol{c} \\ \boldsymbol{c}^{\mathrm{T}} & c(x_*, x_*) \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{1} \\ 1 \end{pmatrix} \\ &= \mathbf{1}^{\mathrm{T}} C_n^{-1} \mathbf{1} + \frac{(1 - \mathbf{1}^{\mathrm{T}} C_n^{-1} \boldsymbol{c})^2}{c(x_*, x_*) - \boldsymbol{c}^{\mathrm{T}} C_n^{-1} \boldsymbol{c}} \end{aligned} \tag{5.19}$$

Which only requires maximizing the second term, since the first term is only dependent on the first n design points. Note that this criterion depends only on our beliefs about the correlation structure of $B(\cdot)$, not the mean and variance of its value at any x.

This criterion is also independent of the previous function evaluations and so contravenes the first of the above criteria for a good design – using all available information – as it does not use any information about the function evaluations when it chooses the next design point. It also contravenes the second, the design eventually covers the whole of the design space, with new points being spread further and further out and does not converge on the zero. If we limit the design space to a large but finite interval of the real line then the criterion wants to repeatedly choose points near the boundary. Using the simple function $f_1(\cdot)$ as a test function, and $\sigma_b^2 = \sigma_0^2 = b = \theta = 1$ and $\mu_0 = 0$ as our prior beliefs we obtain the design in Figure 5.2. We observe that the prevision of X_0 and the design points do not converge to any point, and so this criterion is virtually useless.

5.3.3 The blinkered methods

Like a horse with blinkers on, we could also charge down the path of choosing the most likely place for the zero, by using our current estimate of the zero as a next design point. We could choose the next design point to be the linear estimate of X_0; a "Newtonian" estimate of X_0; or a point where the estimate of $Y(x)$ is equal to zero, replacing the zero finding problem of the expensive function with that of the approximant – "inverse interpolation".

i	x_i	Y_i	$\mathrm{P}_{\mathbf{Y}}(X_0)$
1	0.0000	-0.0998	0.0499
2	-1.3625	-0.7042	-0.0448
3	1.3317	0.5361	0.0538
4	3.0000	0.9854	0.1197
5	-3.0000	-0.9996	0.0515
6	-0.7805	-0.4708	0.0494
7	0.5436	0.1709	0.0527
8	-1.7480	-0.8272	0.0482
9	1.9441	0.7657	0.0519
10	3.0000	0.9854	0.0573
11	2.9947	0.9850	2.8868
12	-0.0000	-0.0998	1.4591
13	-1.9334	-0.8756	2.9544
14	-1.7480	-0.8272	2.5902
15	-0.7805	-0.4708	2.7546

Table 5.2: Sequential design, using naive criterion for $f_1(\cdot)$

The first one again behaves poorly, as should be expected, we are trying to fit a linear estimate to a very non-linear quantity – if we think of a straight-forward linear regression, the location of zero is the ratio of the slope and the intercept. If we look at Table 5.3, we see that this time the design does converge, but not to the zero of the function, but to a value that is a compromise between the zero and our prior belief about X_0.

In equation (5.3) we model $Y(x)$ by $(x - X_0)B(x)$, manipulating this equation gives us $X_0 = x - Y(x)/B(x)$ for each x. Which has a similar form to the Newton Raphson equation $x_{n+1} = x_n - y(x_n)/y'(x_n)$. This "Newtonian" estimate, requires us to estimate $Y(x)/B(x)$ for some x, we would like to use our posterior prevision $\mathrm{P}_{\mathbf{Y}}(Y(x)/B(x))$, but under the Bayes linear framework we do not have this. Instead we can approximate it, assuming all the posterior moments exist for $Y(x)$ and $B(x)$, and that neither of their expectations are zero, we use a bivariate Taylor series expansions of the prevision about $(\mathrm{P}_{\mathbf{Y}}(Y(x)), \mathrm{P}_{\mathbf{Y}}(B(x)))$. The first order expansion

i	x_i	Y_i	$\mathrm{P}_{\boldsymbol{Y}}(X_0)$
1	0.0000	-0.0998	0.0499
2	0.0499	-0.0750	0.0541
3	0.0541	-0.0729	0.0601
4	0.0601	-0.0699	0.0598
5	0.0598	-0.0700	0.0598
6	0.0598	-0.0700	0.0598
7	0.0598	-0.0700	0.0598
8	0.0598	-0.0700	0.0596
9	0.0596	-0.0702	0.0601
10	0.0601	-0.0699	0.0598
11	0.0598	-0.0701	0.0597
12	0.0597	-0.0701	0.0598
13	0.0598	-0.0701	0.0598
14	0.0598	-0.0701	0.0598
15	0.0598	-0.0701	0.0598

Table 5.3: Blinkered design, using posterior prevision of X_0 as the design criterion.

leads to

$$\mathrm{P}_{\boldsymbol{Y}}\left(\frac{Y(x)}{B(x)}\right) \simeq \frac{\mathrm{P}_{\boldsymbol{Y}}(Y(x))}{\mathrm{P}_{\boldsymbol{Y}}(B(x))} \tag{5.20}$$

We can improve on this by using our knowledge about the second-order structure to obtain

$$\mathrm{P}_{\boldsymbol{Y}}\left(\frac{Y(x)}{B(x)}\right) \simeq \frac{\mathrm{P}_{\boldsymbol{Y}}(Y(x))}{\mathrm{P}_{\boldsymbol{Y}}(B(x))}\left(1+\frac{\mathrm{Var}[B(x)/\boldsymbol{Y}]}{\mathrm{P}^2_{\boldsymbol{Y}}(B(x))}-\frac{\mathrm{Cov}([B(x)/\boldsymbol{Y}],[Y(x)/\boldsymbol{Y}])}{\mathrm{P}_{\boldsymbol{Y}}(B(x))\mathrm{P}_{\boldsymbol{Y}}(Y(x))}\right) \tag{5.21}$$

So we can now obtain two estimates for X_0, which we will denote by $\hat{X}_1(x)$ and $\hat{X}_2(x)$,

$$\hat{X}_1(x) = x-\frac{\mathrm{P}_{\boldsymbol{Y}}(Y(x))}{\mathrm{P}_{\boldsymbol{Y}}(B(x))} \tag{5.22}$$

$$\hat{X}_2(x) = x-\frac{\mathrm{P}_{\boldsymbol{Y}}(Y(x))}{\mathrm{P}_{\boldsymbol{Y}}(B(x))}\left(1+\frac{\mathrm{Var}[B(x)/\boldsymbol{Y}]}{\mathrm{P}^2_{\boldsymbol{Y}}(B(x))}-\frac{\mathrm{Cov}([B(x)/\boldsymbol{Y}],[Y(x)/\boldsymbol{Y}])}{\mathrm{P}_{\boldsymbol{Y}}(B(x))\mathrm{P}_{\boldsymbol{Y}}(Y(x))}\right) \tag{5.23}$$

These both simplify if we assume x to be one of the points at which we have evaluated the function, as $Y(x)$ is then a fixed quantity. It is easy to check that the prior

prevision for both these estimates is μ_0. We can also estimate their variances in a similar way, if we look at the prevision of the second order power series expansion of the ratio,

$$\begin{aligned}\mathrm{Cov}(\hat{X}_2(x),\hat{X}_2(x^*)) \simeq \frac{\mathrm{P}_{\boldsymbol{Y}}(Y(x))}{\mathrm{P}_{\boldsymbol{Y}}(B(x))}\frac{\mathrm{P}_{\boldsymbol{Y}}(Y(x^*))}{\mathrm{P}_{\boldsymbol{Y}}(B(x^*))}\Bigg(&\frac{\mathrm{Cov}([Y(x)/\boldsymbol{Y}],[Y(x^*)/\boldsymbol{Y}])}{\mathrm{P}_{\boldsymbol{Y}}(Y(x))\mathrm{P}_{\boldsymbol{Y}}(Y(x^*))}\\ &\frac{\mathrm{Cov}([B(x)/\boldsymbol{Y}],[Y(x^*)/\boldsymbol{Y}])}{\mathrm{P}_{\boldsymbol{Y}}(B(x))\mathrm{P}_{\boldsymbol{Y}}(Y(x^*))}-\frac{\mathrm{Cov}([Y(x)/\boldsymbol{Y}],[B(x^*)/\boldsymbol{Y}])}{\mathrm{P}_{\boldsymbol{Y}}(Y(x))\mathrm{P}_{\boldsymbol{Y}}(B(x^*))}+\\ &\frac{\mathrm{Cov}([B(x)/\boldsymbol{Y}],[B(x^*)/\boldsymbol{Y}])}{\mathrm{P}_{\boldsymbol{Y}}(B(x))\mathrm{P}_{\boldsymbol{Y}}(B(x^*))}\Bigg)\end{aligned} \tag{5.24}$$

whose prior value is $\sigma_0^2(1+\frac{\sigma_b^2}{b^2})$, which is larger than our prior variance for X_0 if we do not have precise information about B, and increases as b gets nearer to zero (as would be expected as the tangent line gets flatter, so a small increase in the y-direction leads to a large increase in the x-direction).

The simple way to use these methods is to mimic Newton-Raphson, we evaluate one of the approximants at our current estimate of X_0, and then use the result as our new estimate of X_0. Here we used these methods on $f_1(\cdot)$, and produced designs listed in Tables 5.4 and 5.5, $\hat{X}_2$ converges much quicker on the zero that $\hat{X}_1$.

i	x_i	$\hat{X}_1(x_i)$	$\hat{X}_2(x_i)$
1	0.0000	0.0998	0.1997
2	0.0998	0.1601	0.2183
3	0.1601	0.1842	0.2075
4	0.1842	0.1938	0.2030
5	0.1938	0.1975	0.2012
6	0.1975	0.1990	0.2005
7	0.1990	0.1996	0.2002
8	0.1996	0.1998	0.2001
9	0.1998	0.1999	0.2000
10	0.1999	0.2000	0.2000

Table 5.4: Design for $f_1(\cdot)$, using $\hat{X}_1(x_i)$ as the next design point.

The way these two approximants behave depends on the function we are approx-

i	x_i	$\hat{X}_1(x_i)$	$\hat{X}_2(x_i)$
1	0.0000	0.0998	0.1997
2	0.1997	0.1999	0.2002
3	0.2002	0.2000	0.2000
4	0.2000	0.2000	0.2000

Table 5.5: Design for $f_1(\cdot)$, using $\hat{X}_2(x_i)$ as the next design point.

imating, $\hat{X}_2(\cdot)$ works well if the function "behaves" i.e. fits our model, whereas $\hat{X}_1(\cdot)$ works (slowly) for most functions we have tried. We can obtain further insight about these predictors by plotting them as a function of X, along with their variances: Figures 5.9 and 5.10 plot the predictor $\mathrm{P}_{\boldsymbol{Y}}(Y(x))$, $\hat{X}_1(x)$ and $\hat{X}_2(x)$ for the first five points of each of the above designs (the latter two with $\pm$1SD lines); and Figures 5.11 and 5.12 plot the covariance between the $\hat{X}_2(x)$s.

A less naive way would be to combine together the estimates at various values of x, for example, by taking a weighted average of the predictors at m points $z_1, \ldots, z_m$

$$\hat{X} = \sum_{i=1}^{m} a_i \hat{X}(z_i)$$

with $\sum_{i=1}^{m} a_i = 1$. The estimate of this with minimum posterior variance has coefficients $\boldsymbol{a} = (a_1, \ldots, a_m)^{\mathrm{T}}$ given by

$$\boldsymbol{a} = \frac{B^{-1}\mathbf{1}}{\mathbf{1}^{\mathrm{T}}B^{-1}\mathbf{1}} \tag{5.25}$$

where $B_{ij} = \mathrm{Cov}(\hat{X}(z_i), \hat{X}(z_j))$.

The third method is to use the adjusted prevision $\mathrm{P}_{\boldsymbol{Y}}(Y(x))$ as an approximation to the function, and then find values of x which solve the simpler problem $\mathrm{P}_{\boldsymbol{Y}}(Y(x)) = 0$. This has the advantage that the approximation is no longer linear, but uses all the detail of the approximant. It has at least two disadvantages, firstly we still have to solve the possibly complicated numerical inverse interpolation problem, which is further complicated by the second, that there might be many zeros, several

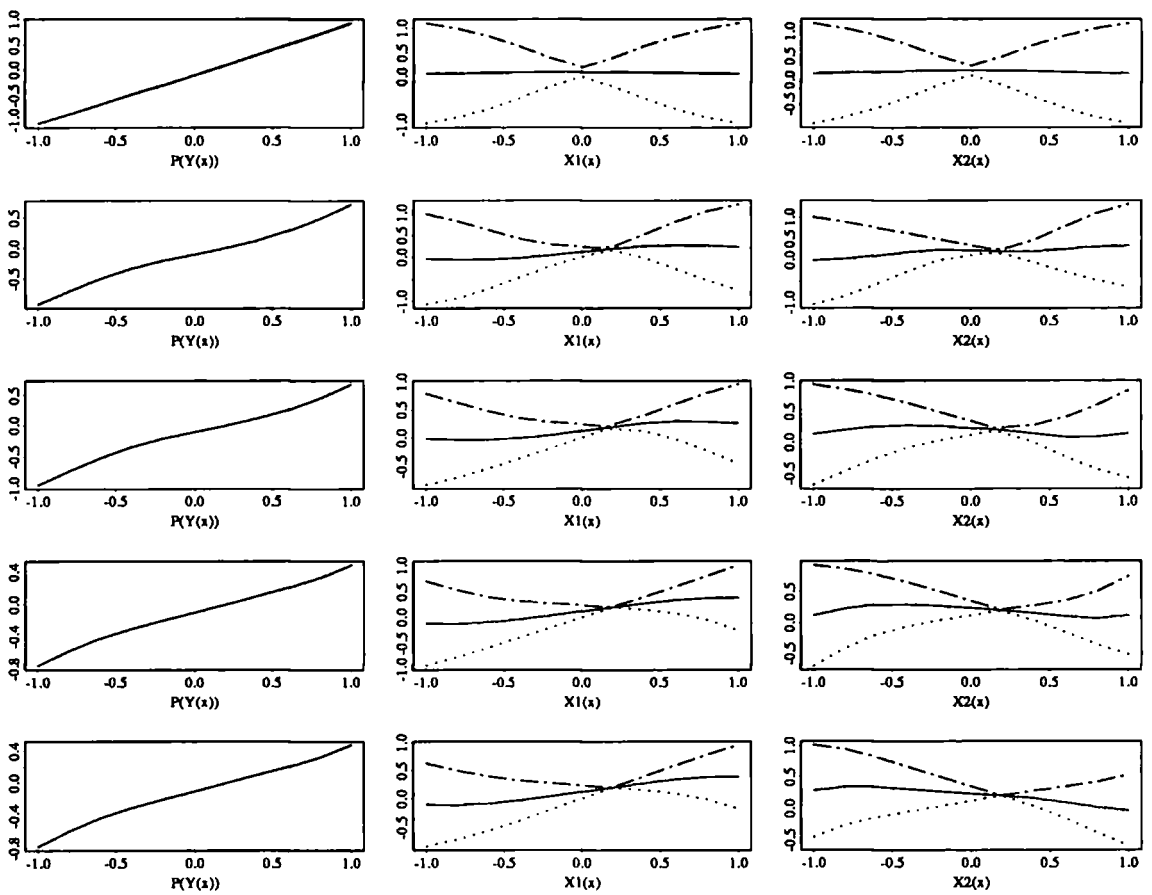

Figure 5.9: $\mathrm{P}_{\mathbf{Y}}(Y(x))$, $\hat{X}_1(x)$ and $\hat{X}_2(x)$ after one to five design points, using $\hat{X}_1(\cdot)$ to choose the next design point.

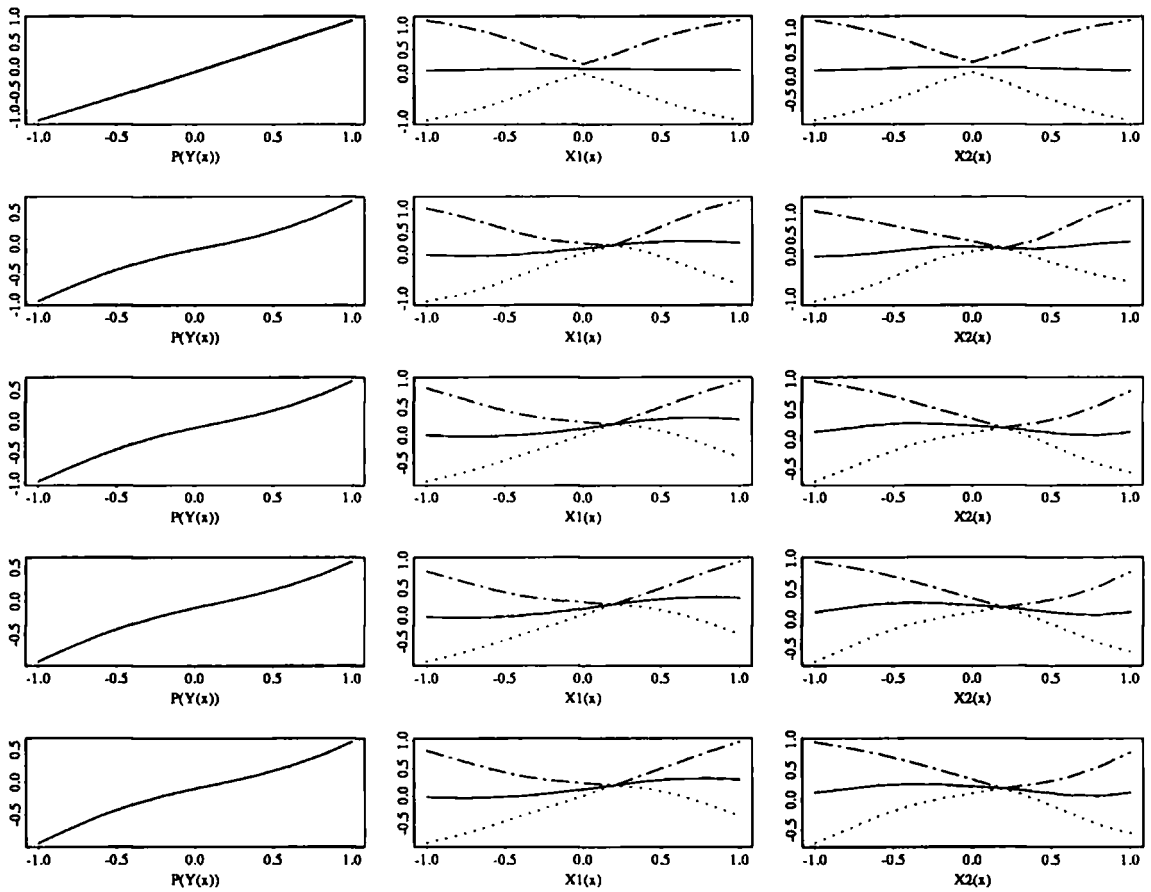

Figure 5.10: $P_{\mathbf{Y}}(Y(x))$, $\hat{X}_1(x)$ and $\hat{X}_2(x)$ after one to five design points, using $\hat{X}_2(\cdot)$ to choose the next design point.

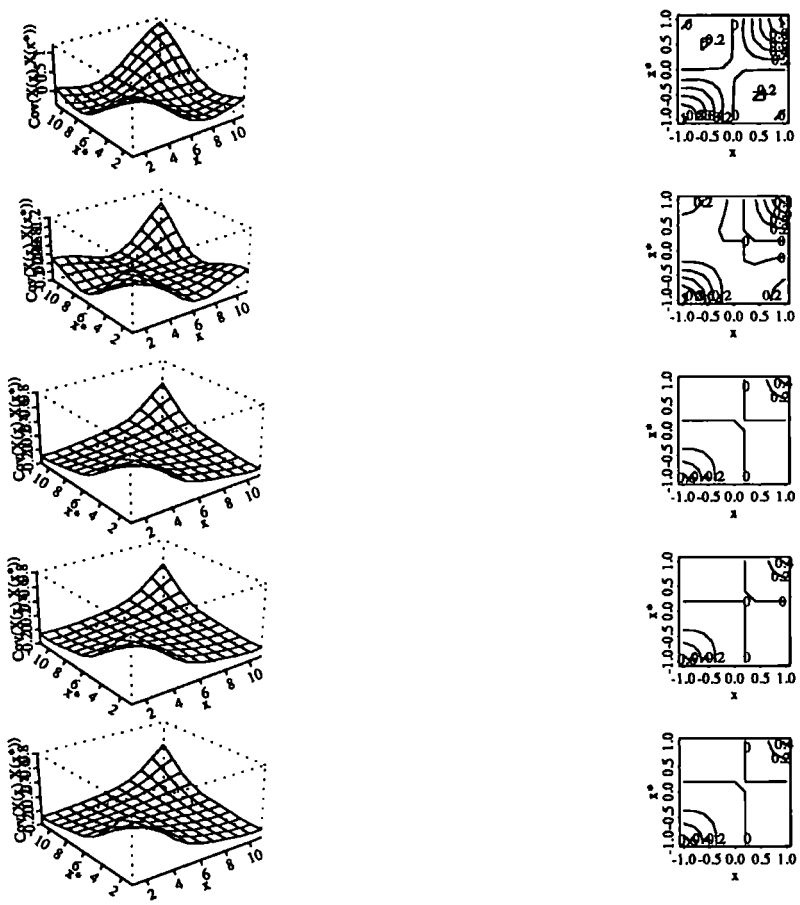

Figure 5.11: Covariance between the $\hat{X}_2(x)$s after one to five design points, using $\hat{X}_1(\cdot)$ to choose the next design point.

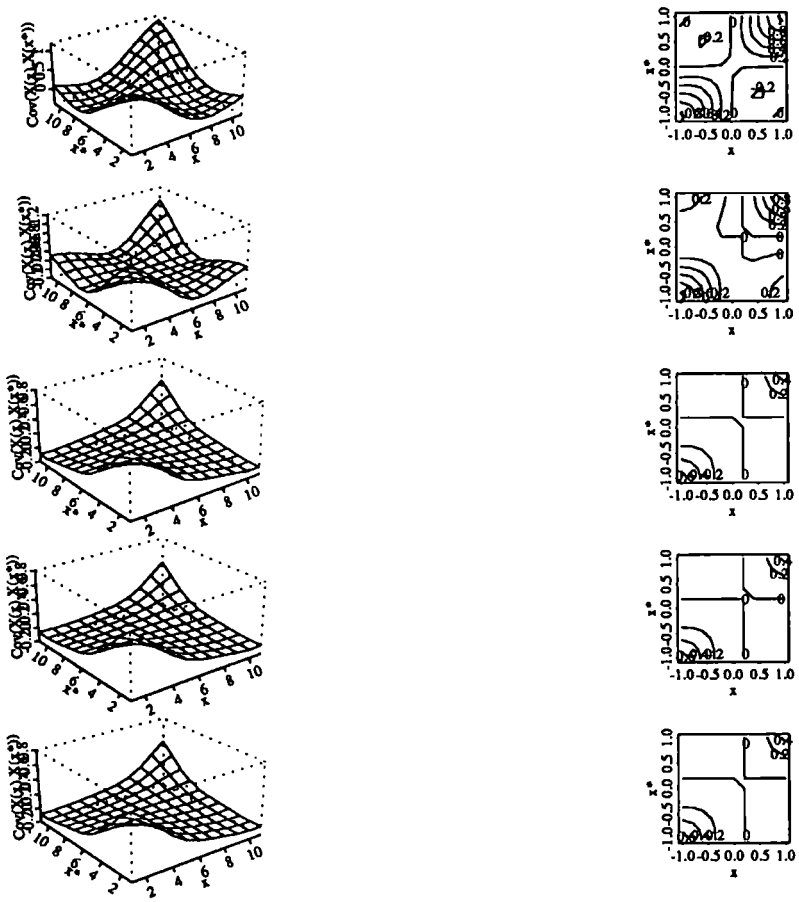

Figure 5.12: Covariance between the $\hat{X}_2(x)$s after one to five design points, using $\hat{X}_2(\cdot)$ to choose the next design point.

of which are caused by oscillations in the interpolant.

We will again apply this method to our simple example $f_1(\cdot)$, this locates the zero in just five function evaluations, summarised in Table 5.6 and Figure 5.13 below. In this case the approximant has just one zero.

i	x_i	$Y(x_i)$	Zero of $\mathrm{P}_{\mathbf{Y}}(Y(\cdot))$	$Y(\cdot)$ at 'zero'	$\sqrt{\mathrm{Var}[Y(\cdot)/\mathbf{Y}]}$ at 'zero'
1	0.000000	-0.099833	0.099343	-0.050307	0.171645
2	0.099343	-0.050307	0.195665	-0.002168	0.039596
3	0.195665	-0.002168	0.199912	-0.000044	0.000155
4	0.199912	-0.000044	0.200000	0.000000	0.000000
5	0.200000	0.000000	0.200000	0.000000	0.000000

Table 5.6: "Inverse Interpolation" design for $f_1(\cdot)$, with model parameters $\theta = b = \sigma_0^2 = \sigma_b^2 = 1$ and $\mu_0 = 0$

All these methods have two main disadvantages, firstly they can be lead down, the proverbial, garden path – if the function dips sharply but does not cross the axis at th minimum, the design will tend to cluster around this minimum, and not be able to escape. Secondly the methods also always cluster the points together even when not converging on a "false zero", and so the columns of the correlation matrix become very similar, and hence it becomes ill-conditioned (see Section 5.12).

5.3.4 Variance modified criteria

Instead of looking for values of x which solve the inverse interpolation problem $P_{\mathbf{Y}}(Y(x)) = 0$, we could look for values of x which minimize its squared prevision. In the linear framework we are unable to get a true estimate of this, but we can use the adjusted variance estimates to get an approximation to it

$$C_1(x) \stackrel{\text{def}}{=} \mathrm{P}_{\mathbf{Y}}(Y(x)^2) \simeq P^2_{\mathbf{Y}}(Y(x)) + \mathrm{Var}[Y(x)/\mathbf{Y}] \tag{5.26}$$

We then find values of x to minimize $C_1(\cdot)$. We can also use modifications of this criterion, which give more weight to points where the squared prevision is small in comparison to our uncertainty about its value there. We can do this by dividing

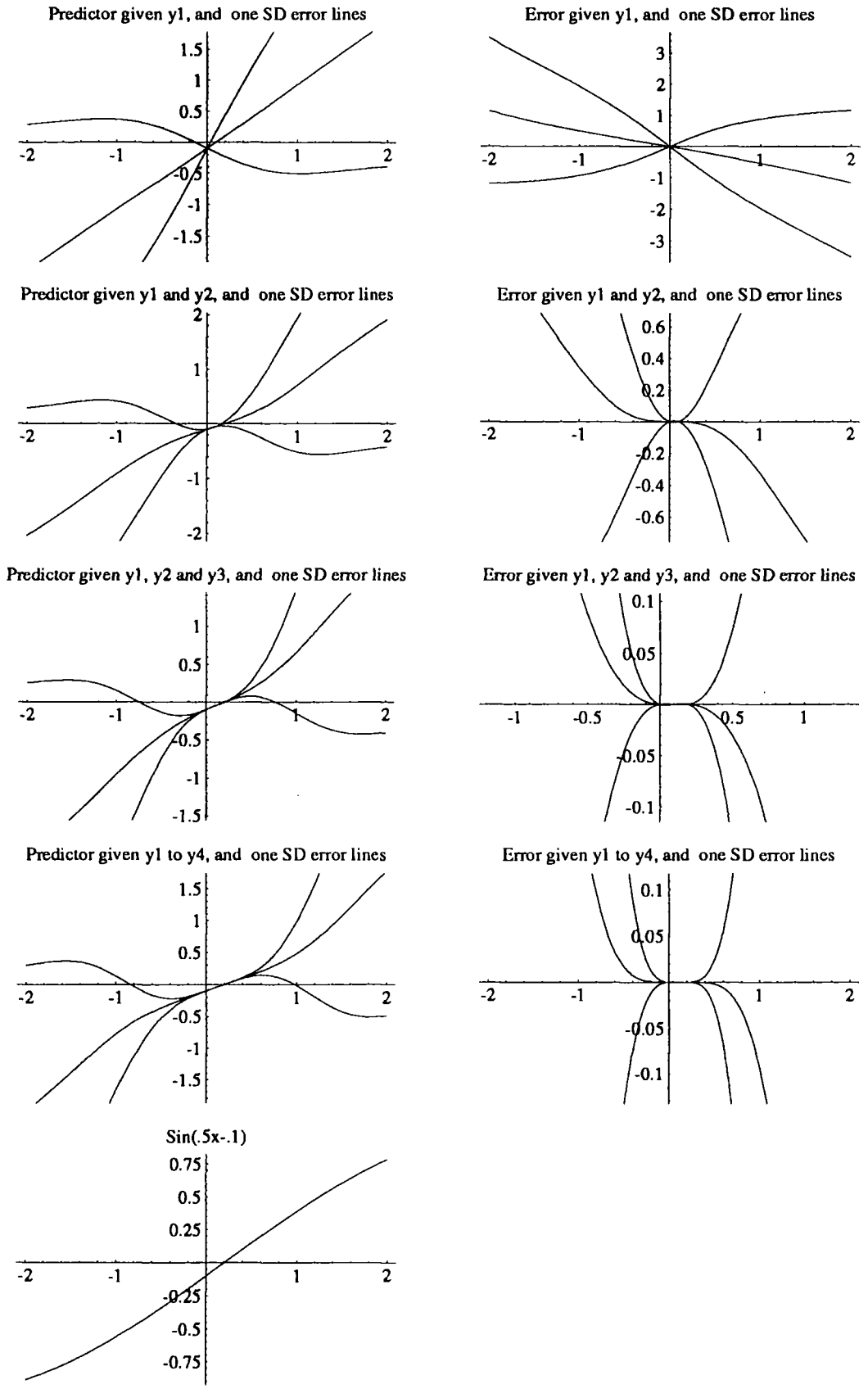

Figure 5.13: "Inverse Interpolation" design for $f_1(\cdot)$, its approximant and errors using function evaluations at $x_1 = 0$, $x_2 = 0.099343$, $x_3 = 0.195665$ and $x_4 = 0.199912$, with model parameters $\theta = b = \sigma_0^2 = \sigma_b^2 = 1$ and $\mu = 0$

through by our prior or adjusted variance to obtain,

$$C_2(x) \stackrel{\text{def}}{=} \frac{\mathrm{P}_{\boldsymbol{Y}}(Y(x)^2)}{\mathrm{Var}(Y(x))} \simeq \frac{P^2_{\boldsymbol{Y}}(Y(x)) + \mathrm{Var}[Y(x)/\boldsymbol{Y}]}{\mathrm{Var}(Y(x))} \tag{5.27}$$

$$C_3(x) \stackrel{\text{def}}{=} \frac{\mathrm{P}_{\boldsymbol{Y}}(Y(x)^2)}{\mathrm{Var}[Y(x)/\boldsymbol{Y}]} - 1 \simeq \frac{P^2_{\boldsymbol{Y}}(Y(x))}{\mathrm{Var}[Y(x)/\boldsymbol{Y}]} \tag{5.28}$$

The last criteria is equivalent to the "inverse interpolation" method of Section 5.3.3. Minimizing C_1 and C_2 appear to work well, often producing very similar sequential designs, because the prior variance does not change much over the region where the zero is. If the function has multiple zeros the design may flip between two points, so highlighting this fact.

To optimize C_1 and C_2 we assume that the criterion is unimodal between the design points, and use the "Golden-Section search" algorithm to locate the unique local minimum in each interval, choosing the least of these as the global minimum. For every function we have examined this appears to be the case, although we have no analytic proof of this fact.

We now try these methods out on functions $f_1(\cdot)$ and $f_2(\cdot)$, with our standard prior beliefs $\sigma_0^2 = \sigma_b^2 = \theta = b = 1$ and $\mu_0 = 0$. The designs generated are listed in Tables 5.7 to 5.10. We observe that there is very little difference in the number of steps needed to find the zero in either case. (The odd design point (0.1215) in Table 5.9 is probably due to numerical errors in the inversion of the covariance matrix)

i	x_i	Y_i
1	0.0000	-0.0998
2	0.0250	-0.0874
3	0.1358	-0.0321
4	0.1964	-0.0018
5	0.2000	-0.0000

Table 5.7: Design generated by criterion C_1 for function $f_1(\cdot)$ with $\sigma_0^2 = \sigma_b^2 = \theta = b = 1$ and $\mu_0 = 0$

The small amount of extra work with this method in comparison to the "inverse

i	x_i	Y_i
1	0.0000	-0.0998
2	0.0250	-0.0874
3	0.1360	-0.0320
4	0.1968	-0.0018
5	0.2000	-0.0000

Table 5.8: Design generated by criterion C_2 for function $f_1(\cdot)$ with $\sigma_0^2 = \sigma_b^2 = \theta = b = 1$ and $\mu_0 = 0$

i	x_i	Y_i
1	0.0000	0.0198
2	-0.0197	0.0198
3	-0.2804	0.0195
4	-0.4077	0.0156
5	-0.5803	-0.0455
6	-0.4866	0.0038
7	-0.4995	0.0001
8	0.1215	0.0198
9	-0.5000	0.0000

Table 5.9: Design generated by criterion C_1 for function $f_2(\cdot)$ with $\sigma_0^2 = \sigma_b^2 = \theta = b = 1$ and $\mu_0 = 0$

i	x_i	Y_i
1	0.0000	0.0198
2	-0.0049	0.0198
3	-0.0172	0.0198
4	-0.1757	0.0198
5	-0.2872	0.0194
6	-0.5195	-0.0070
7	-0.4943	0.0170
8	-0.5000	-0.0000

Table 5.10: Design generated by criterion C_2 for function $f_2(\cdot)$ with $\sigma_0^2 = \sigma_b^2 = \theta = b = 1$ and $\mu_0 = 0$

interpolation" methods above pays dividends, as it has some advantages. Firstly it is less inclined to move a long way away from previous function evaluations to areas of the function we are unsure about. If the function being approximated does not behave well or some of the parameters have been mis-specified, the approximant can oscillate wildly, cutting the axis in many places generally away from previous design points. These criteria C_1 and C_2 try to ignore these. Secondly, it usually produces a unique global minimum, whereas the "inverse interpolation" often finds multiple zero.

5.3.5 Modifying the first order structure

If we look at the form of equation (5.18), we note that the predictor contains $\mathrm{P}_{\boldsymbol{Y}}(X_0)$ as an estimate of X_0 in it, but there is not an updated estimate for b, it appears as b. We could therefore make an *ad hoc* modification to this predictor, by replacing b by an estimate of the slope – one possibility is to replace b by $\mathrm{P}_{\boldsymbol{Y}}(\cdot)$ for our current estimate of X_0, ie $\mathrm{P}_{\boldsymbol{Y}}(B(\mathrm{P}_{\boldsymbol{Y}}(X_0)))$. We can then use this modification in a similar way to the methods in Section 5.3.3.

5.3.6 Trying to modify the second order structure

As we only specify the first and second order structure of the function we have no way to estimate the variance of the predictor from the data using Bayes linear methodology. We can again use *ad hoc* methods to estimate the product of σ_b^2 and σ_0^2, we cannot get at the two separately, if we put $s^2 = \sigma_0^2\sigma_b^2$

$$\hat{s}^2 = \frac{(\boldsymbol{x} - \frac{\boldsymbol{Y}}{b} - \hat{\mu}\mathbf{1})^{\mathrm{T}} C^{-1} (\boldsymbol{x} - \frac{\boldsymbol{Y}}{b} - \hat{\mu}\mathbf{1}) - \frac{(\mathbf{1}^{\mathrm{T}} C^{-1}(\boldsymbol{x} - \frac{\boldsymbol{Y}}{b} - \hat{\mu}\mathbf{1}))^2}{\mathbf{1}^{\mathrm{T}} C^{-1} \mathbf{1}}}{b^2(n-1)} \tag{5.29}$$

where $\hat{\mu} = \mathbf{1}^{\mathrm{T}}(\boldsymbol{x} - \frac{\boldsymbol{Y}}{b})/n$. We then define $\hat{\sigma}_b^2$ and $\hat{\sigma}_0^2$ so their product is $\hat{s}^2$, and there ratio is the same as our adjusted variances for $\mathrm{Var}[X_0/\boldsymbol{Y}]$ and $\mathrm{Var}[B(X_0)/\boldsymbol{Y}]$.

This method seems contrived, and in general their is no real improvement on the

rate at which the method finds the zero.

5.3.7 Initial exploration designs

It would be desirable to have a criterion like that in Schagen[1984], which is a compromise between exploring the region and looking for the zero of the function. We have examined the second part above. One possibility for an exploratory criterion, is to choose the design point to minimize the average (or maximum) variance of the approximant over the design space. Another possibility is to minimize the average (or maximum) ratio of posterior to prior variances, Goldstien's $D(Y)$ defined in Section 3.5. We will look at another similar criterion in the next section.

If we have two criteria, scaled so they are of a similar magnitude, a zero-finding one $\mathcal{C}_0(\cdot)$ and an exploratory one $\mathcal{C}_E(\cdot)$, we can combine the two to form a composite criterion

$$\mathcal{C}(\cdot) = W\mathcal{C}_0(\cdot) + (1 - W)\mathcal{C}_E(\cdot)$$

where initially W is near 0 and the criterion selects a mainly exploratory design, and W eventually increases to be near 1 and the design then homes in on the zero.

5.3.8 Belief grid criteria

We can use a criteria similar to that in Chapter 4 to explore the region, the criteria chooses points (or a point) from a grid to minimize the trace of the belief transform. Because we are using a grid based design, we can, after each stage, easily modify our grid to reflect where we think the zero is. This achieves the objectives of the previous section: initially we have very vague knowledge where the zero is and so the grid is spread out, but as our knowledge of the function increases, and we have more information about the location of the zero, then we can refine the grid to include more points near where we think the zero is at the moment.

We can use this method to produce small pre-determined initial designs. For example, if we are looking for a three point initial design, we would lay down a grid,

for example $\{-1, -0.9, \ldots, 0.9, 1\}$, and our prior beliefs and compute the transform. In this case (with our standard prior beliefs $\theta = b = \sigma_b^2 = \sigma_0^2 = 1$, $\mu_0 = 0$) we have the optimal three point design, $\{-0.3, 0, 0.3\}$. On the same grid the optimal two point design is $\pm\{-0.2, 0.3\}$.

5.4 Choosing the smoothness parameter

An important question that so far has not been addressed is the value of θ, the correlation parameter in the model that controls the smoothness of $Y(\cdot)$. We can elicit an initial value for θ using ideas put forward in Section 5.7, but we would like some way of estimating it, not necessarily using Bayes linear methods.

We can use cross validation (a non-Bayesian method) to choose a "best" value of θ, as follows. For all the design points x_i used so far, we compute an estimate $\hat{y}_i$ for $y(x_i)$, based on the other design points and then compare this estimate with the true function value. We define the *Cross Validation Mean Square Error* as

$$CVMSE(\theta) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y(x_i))^2$$

which is an average measure of this deviation. We then choose the value of θ that minimizes this.

For example, if we were approximating $f_1(\cdot)$ by function evaluations at -0.5, 0 and 0.5, we can plot the $CVMSE$ as a function of θ, in Figure 5.14, using our standard prior beliefs $\sigma_0^2 = \sigma_b^2 = b = 1$, $\mu_0 = 0$, the optimal value of θ is 0.00283. This will quite often be an underestimate of θ, as when there are only a handful of points, we can always fit a much smoother curve through them. If we extend the design to include two extra points at ± 0.25, we can re plot the $CVMSE$ curve, due to rounding errors – which creep in because the covariance matrix is very ill-conditioned for small values of θ –the function no-longer has a unique minimum, but if we look at the bottom curve of Figure 5.15, we can see that the $CVMSE$ is now minimized by $\theta \simeq 0.0145$, which is five times greater than the value we got for the three point

problem, suggesting the curve we are now fitting is less smooth, as would be expected. The problem of numerical rounding errors is even more accentuated when we apply this method to some of the optimal designs examined in the previous section, as they tend to place points closer together, and so smoother functions can be generally be fitted through them, and so the optimal θ value is often very small.

5.5 An extension to the model

We can extend the previous model by assuming that the average of the slope process is an unknown quantity B, with a given mean and variance. We therefore modify the assumptions we made in the previous section as follows

$$\mathrm{P}_B(B(x)) = B \tag{5.30}$$

$$\mathrm{Cov}([B(x)/B],[B(x_*)/B]) = \sigma_b^2\rho(x-x_*) \tag{5.31}$$

where this additional variance term is independent of B, and where

$$\mathrm{P}(B) = b \tag{5.32}$$

$$\mathrm{Var}(B) = \sigma_M^2 \tag{5.33}$$

These two additional assumptions give us the following second order structure for $B(\cdot)$:

$$\begin{aligned} \mathrm{P}(B(x)) &= \mathrm{P}(\mathrm{P}_B(B(x))) \\ &= \mathrm{P}(B) = b \end{aligned} \tag{5.34}$$

$$\begin{aligned} \mathrm{Cov}(B(x),B(x_*)) &= P(B(x)B(x_*)) - P(B(x))P(B(x_*)) \\ &= P((B(x)-B)(B(x_*)-B) + P(B^2) - P(B)^2 \\ &= \sigma_b^2\rho(x-x_*) + \sigma_M^2 \end{aligned} \tag{5.35}$$

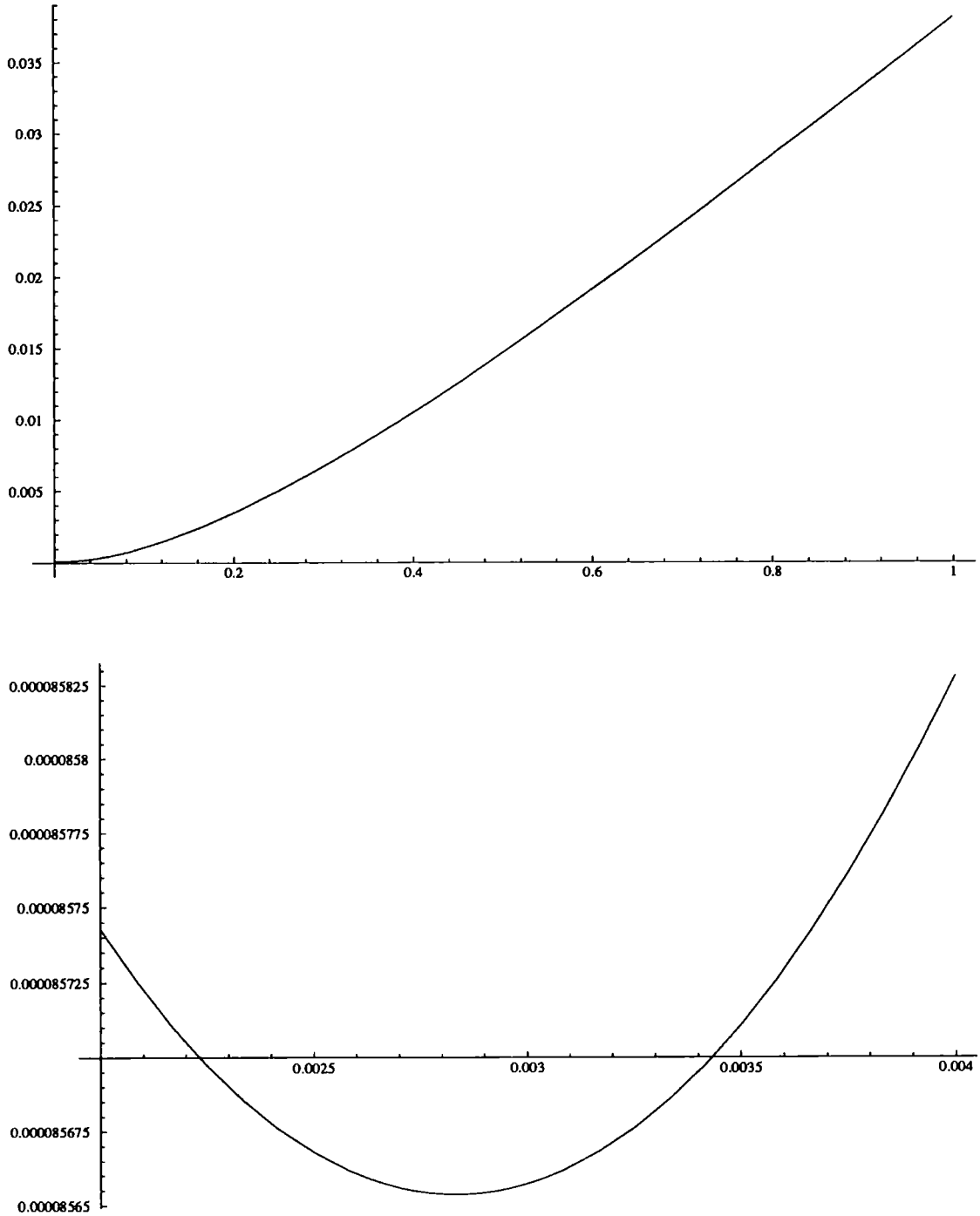

Figure 5.14: Plot of $CVMSE$ against θ, with the design $\{-0.5, 0, 0.5\}$

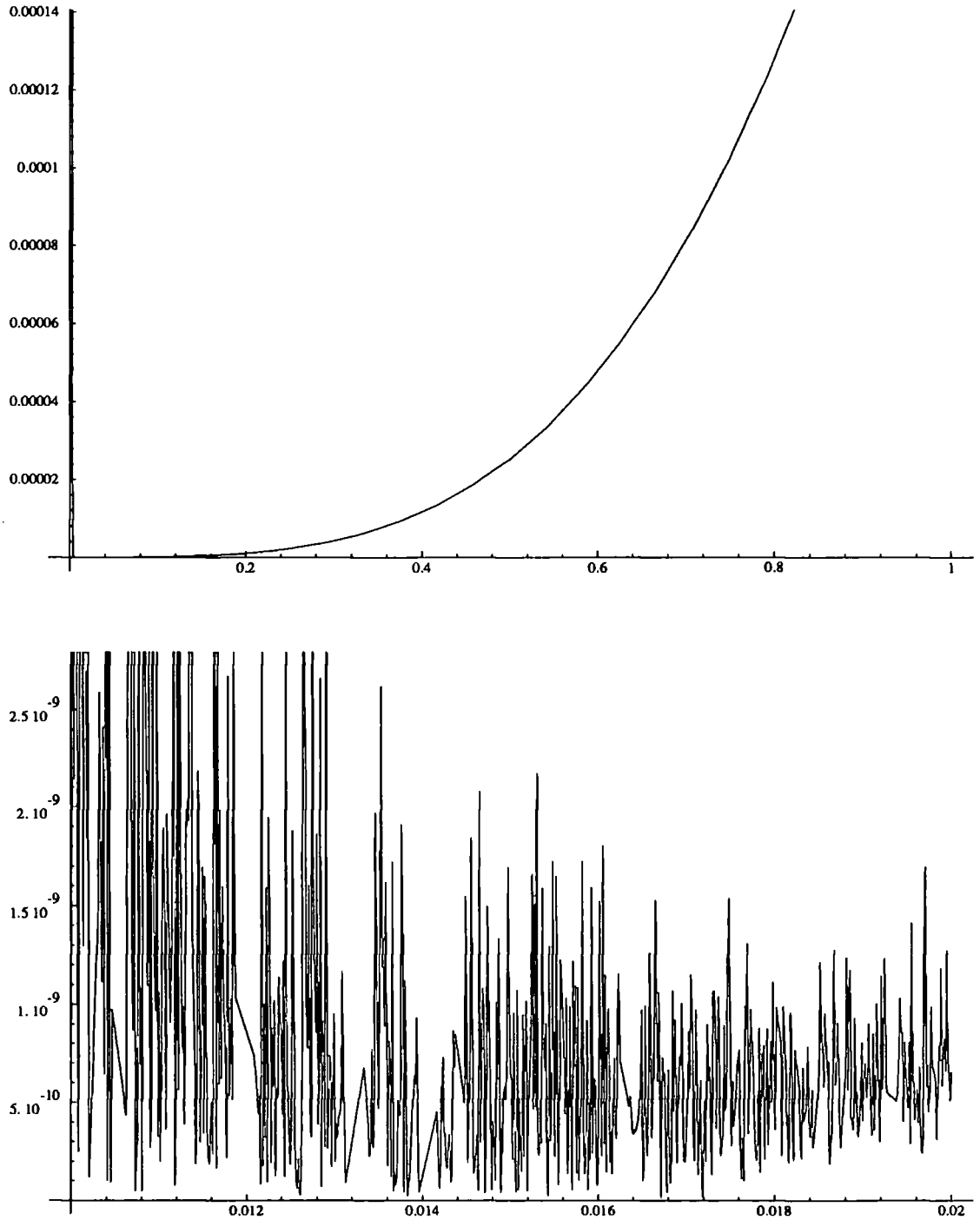

Figure 5.15: Plot of $CVMSE$ against θ, with the design $\{-0.5, -0.25, 0, 0.25, 0.5\}$

This can be written in the same form as the model in the Section 5.2, i.e. $\hat{\sigma}_b^2\hat{\rho}(x - x_*)$, with $\hat{\sigma}_b^2 = \sigma_b^2 + \sigma_M^2$ and $\hat{\rho}(x - x_*) = \frac{\sigma_M^2 + \sigma_b^2\rho(x-x_*)}{\sigma_M^2+\sigma_b^2}$, and so no further algebraic work has to be done to work out the estimates.

We can now plead "ignorance" for the value of the slope parameter B, by letting its prior variance σ_M^2 tend to infinity, and obtain posterior estimates for the prevision and variance of B, $B(\cdot)$ and $Y(\cdot)$. In this limiting case, the posterior estimates for the mean and variance for X_0 are the same as their prior values – we can learn nothing about the position of the zero from the data, due to the infinite variance about B.

If we take observations $Y_1, \ldots, Y_n$ at $x_1, \ldots, x_n$ we can write down the covariance matrices.

$$\begin{aligned}
\mathrm{Var}(\boldsymbol{Y}, \boldsymbol{Y}) &= \sigma_M^2(\boldsymbol{x} - \mu_0\mathbf{1})(\boldsymbol{x} - \mu_0\mathbf{1})^{\mathrm{T}} + (b^2 + \sigma_M^2)\sigma_0^2\mathbf{1}\mathbf{1}^{\mathrm{T}} + \sigma_b^2 C \\
\mathrm{Cov}(Y(x_*), \boldsymbol{Y}) &= \sigma_M^2(x_* - \mu_0)(\boldsymbol{x} - \mu_0\mathbf{1})^{\mathrm{T}} + (b^2 + \sigma_M^2)\sigma_0^2\mathbf{1}^{\mathrm{T}} + \sigma_b^2\boldsymbol{c}^{\mathrm{T}} \\
\mathrm{Cov}(Y(\tilde{x}_*), \boldsymbol{Y}) &= \sigma_M^2(\tilde{x}_* - \mu_0)(\boldsymbol{x} - \mu_0\mathbf{1})^{\mathrm{T}} + (b^2 + \sigma_M^2)\sigma_0^2\mathbf{1}^{\mathrm{T}} + \sigma_b^2\tilde{\boldsymbol{c}}^{\mathrm{T}} \qquad (5.36) \\
\mathrm{Cov}(B(x_*), \boldsymbol{Y}) &= \sigma_M^2(\boldsymbol{x} - \mu_0\mathbf{1})^{\mathrm{T}} + \sigma_b^2\boldsymbol{d}^{\mathrm{T}} \\
\mathrm{Cov}(B(\tilde{x}_*), \boldsymbol{Y}) &= \sigma_M^2(\boldsymbol{x} - \mu_0\mathbf{1})^{\mathrm{T}} + \sigma_b^2\tilde{\boldsymbol{d}}^{\mathrm{T}} \\
\mathrm{Cov}(B, \boldsymbol{Y}) &= \sigma_M^2(\boldsymbol{x} - \mu_0\mathbf{1})^{\mathrm{T}} \qquad (5.37)
\end{aligned}$$

where $\boldsymbol{c}$, $\tilde{\boldsymbol{c}}$, C, $\boldsymbol{d}$ and $\tilde{\boldsymbol{d}}$ are defined exactly as in Section 5.2.

When we let σ_M^2 tend to infinity, even though the prior covariance structure is infinite, the posterior structure is finite and so we are able to compute our posterior previsions of $Y(x_*)$, $B(x_*)$ and B. We will first introduce additional notation, $F = (\mathbf{1}\ \boldsymbol{x})$, $\boldsymbol{f} = (1\ x_*)^{\mathrm{T}}$, $\tilde{\boldsymbol{f}} = (1\ \tilde{x}_*)^{\mathrm{T}}$.

$$\begin{aligned}
\mathrm{P}_{\boldsymbol{Y}}(Y(x_*)) &= \frac{\begin{vmatrix} F^{\mathrm{T}}C^{-1}F & F^{\mathrm{T}}C^{-1}\boldsymbol{Y} \\ \boldsymbol{c}^{\mathrm{T}}C^{-1}F - \boldsymbol{f}^{\mathrm{T}} & \boldsymbol{c}^{\mathrm{T}}C^{-1}\boldsymbol{Y} \end{vmatrix}}{\left|F^{\mathrm{T}}C^{-1}F\right|} \\
&= \boldsymbol{f}^{\mathrm{T}}\boldsymbol{\alpha} + \boldsymbol{c}^{\mathrm{T}}C^{-1}(\boldsymbol{Y} - F\boldsymbol{\alpha}) \qquad (5.38)
\end{aligned}$$

$$\begin{aligned} \mathrm{P}_{\boldsymbol{Y}}(B(x_*)) &= \frac{\begin{vmatrix} F^{\mathrm{T}}C^{-1}F & F^{\mathrm{T}}C^{-1}\boldsymbol{Y} \\ \boldsymbol{d}^{\mathrm{T}}C^{-1}F - (0\ 1) & \boldsymbol{d}^{\mathrm{T}}C^{-1}\boldsymbol{Y} \end{vmatrix}}{|F^{\mathrm{T}}C^{-1}F|} \\ &= \alpha_2 + \boldsymbol{d}^{\mathrm{T}}C^{-1}(\boldsymbol{Y} - F\boldsymbol{\alpha}) \qquad (5.39) \\ \mathrm{P}_{\boldsymbol{Y}}(B) &= \frac{\left|F^{\mathrm{T}}C^{-1}(\ \mathbf{1}\ \ \boldsymbol{Y}\ )\right|}{|F^{\mathrm{T}}C^{-1}F|} \\ &= \alpha_2 \qquad (5.40) \end{aligned}$$

where $\boldsymbol{\alpha} = (F^{\mathrm{T}}C^{-1}F)^{-1}F^{\mathrm{T}}C^{-1}\boldsymbol{Y}$ is the weighted least squares estimate in the linear regression with error covariances given by $c(\cdot,\cdot)$, and their adjusted covariances

$$\begin{aligned} &\mathrm{Cov}([Y(x_*)/\boldsymbol{Y}],[Y(\tilde{x}_*)/\boldsymbol{Y}]) \\ &= \sigma_b^2 c_* - \sigma_b^2 \frac{\begin{vmatrix} F^{\mathrm{T}}C^{-1}F & F^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{c}} - \tilde{\boldsymbol{f}} \\ \boldsymbol{c}^{\mathrm{T}}C^{-1}F - \boldsymbol{f}^{\mathrm{T}} & \boldsymbol{c}^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{c}} \end{vmatrix}}{\left|F^{\mathrm{T}}C^{-1}F\right|} \qquad (5.41) \\ &= \sigma_b^2\left(c(x_*,\tilde{x_*}) + (\boldsymbol{c}^{\mathrm{T}}C^{-1}F - \boldsymbol{f}^{\mathrm{T}})(F^{\mathrm{T}}C^{-1}F)^{-1}(F^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{c}} - \tilde{\boldsymbol{f}}) - \boldsymbol{c}^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{c}}\right) \\ &\mathrm{Cov}([B(x_*)/\boldsymbol{Y}],[B(\tilde{x_*})/\boldsymbol{Y}]) \\ &= \sigma_b^2\rho(x_* - \tilde{x_*}) - \sigma_b^2 \frac{\begin{vmatrix} F^{\mathrm{T}}C^{-1}F & F^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{d}} - (0\ 1)^{\mathrm{T}} \\ \boldsymbol{d}^{\mathrm{T}}C^{-1}F - (0\ 1) & \boldsymbol{d}^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{d}} \end{vmatrix}}{|F^{\mathrm{T}}C^{-1}F|} \\ &= \sigma_b^2\Big(\rho(x_* - \tilde{x_*}) - \boldsymbol{d}^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{d}} + \\ &\qquad [\boldsymbol{d}^{\mathrm{T}}C^{-1}F - (0\ 1)](F^{\mathrm{T}}C^{-1}F)^{-1}\left[F^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{d}} - (0\ 1)^{\mathrm{T}}\right]\Big) \qquad (5.42) \\ &\mathrm{Cov}([B(x_*)/\boldsymbol{Y}],[Y(\tilde{x_*})/\boldsymbol{Y}]) \\ &= \sigma_b^2(\tilde{x_*} - \mu_0)\rho(x_* - \tilde{x_*}) - \sigma_b^2 \frac{\begin{vmatrix} F^{\mathrm{T}}C^{-1}F & F^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{c}} - \tilde{\boldsymbol{f}} \\ \boldsymbol{d}^{\mathrm{T}}C^{-1}F - (0\ 1) & \boldsymbol{d}^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{c}} \end{vmatrix}}{|F^{\mathrm{T}}C^{-1}F|} \\ &= \sigma_b^2\Big((\tilde{x_*} - \mu_0)\rho(x_* - \tilde{x_*}) - \boldsymbol{d}^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{c}} + \\ &\qquad (\boldsymbol{d}^{\mathrm{T}}C^{-1}F - (0\ 1))(F^{\mathrm{T}}C^{-1}F)^{-1}(F^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{c}} - \tilde{\boldsymbol{f}})\Big) \qquad (5.43) \end{aligned}$$

$\mathrm{Var}[B/\boldsymbol{Y}]$

$$= \sigma_b^2 \frac{\left|1^T C^{-1} 1\right|}{\left|F^T C^{-1} F\right|}$$

$$= \sigma_b^2 \left[x^T C^{-1} x - \frac{(x^T C^{-1} 1)^2}{1^T C^{-1} 1} \right]^{-1} \quad (5.44)$$

$$\mathrm{Cov}([B/Y], [Y(\tilde{x}_*)/Y])$$

$$= \frac{\sigma_b^2 \left|F^T C^{-1} 1 \quad F^T C^{-1} \tilde{c} - \tilde{f})\right|}{\left|F^T C^{-1} F\right|}$$

$$= (0 \;\; 1)(F^T C^{-1} F)^{-1}(F^T C^{-1} \tilde{c} - \tilde{f}) \quad (5.45)$$

We can then use these estimates in a similar way to those found in Section 5.3, by using the variance modified criteria, or the "Newtonian" estimation method. We need an initial two point design, and we will use $\{-0.3, 0.3\}$, as this is a near-optimal, two point, symmetric design using the grid based design criteria from Section 5.3.8. For example, for $f_1(\cdot)$, using the variance modified criterion C_1, we have the design listed in Table 5.11; using C_2, we have Table 5.12; and using the "inverse interpolation" method, we have Table 5.13.

i	x_i	Y_i
1	-0.3000	-0.2474
2	0.3000	0.4998
3	0.2847	0.0423
4	0.2062	0.0031
5	0.2000	0.0000

Table 5.11: Design generated by criterion C_1 for function $f_1(\cdot)$ with $\sigma_0^2 = \sigma_b^2 = \theta = 1$ and $\mu_0 = 0$, using the extendend model.

For $f_2(\cdot)$, using C_1, we have Table 5.14; using C_2, we have Table 5.15; and using the "inverse interpolation" method, we have Table 5.16 – this latter is an example of multiple "phantom" zeros and so does not converge.

$$= \sigma_b^2 \frac{\left|\mathbf{1}^T C^{-1} \mathbf{1}\right|}{\left|F^T C^{-1} F\right|}$$

$$= \sigma_b^2 \left[\boldsymbol{x}^T C^{-1} \boldsymbol{x} - \frac{(\boldsymbol{x}^T C^{-1} \mathbf{1})^2}{\mathbf{1}^T C^{-1} \mathbf{1}}\right]^{-1} \tag{5.44}$$

$$\mathrm{Cov}([B/\boldsymbol{Y}], [Y(\tilde{x}_*)/\boldsymbol{Y}])$$

$$= \frac{\sigma_b^2 \left|F^T C^{-1} \mathbf{1} \quad F^T C^{-1} \tilde{\boldsymbol{c}} - \tilde{\boldsymbol{f}})\right|}{\left|F^T C^{-1} F\right|}$$

$$= (0 \;\; 1)(F^T C^{-1} F)^{-1}(F^T C^{-1} \tilde{\boldsymbol{c}} - \tilde{\boldsymbol{f}}) \tag{5.45}$$

We can then use these estimates in a similar way to those found in Section 5.3, by using the variance modified criteria, or the "Newtonian" estimation method. We need an initial two point design, and we will use $\{-0.3, 0.3\}$, as this is a near-optimal, two point, symmetric design using the grid based design criteria from Section 5.3.8. For example, for $f_1(\cdot)$, using the variance modified criterion C_1, we have the design listed in Table 5.11; using C_2, we have Table 5.12; and using the "inverse interpolation" method, we have Table 5.13.

i	x_i	Y_i
1	-0.3000	-0.2474
2	0.3000	0.4998
3	0.2847	0.0423
4	0.2062	0.0031
5	0.2000	0.0000

Table 5.11: Design generated by criterion C_1 for function $f_1(\cdot)$ with $\sigma_0^2 = \sigma_b^2 = \theta = 1$ and $\mu_0 = 0$, using the extended model.

For $f_2(\cdot)$, using C_1, we have Table 5.14; using C_2, we have Table 5.15; and using the "inverse interpolation" method, we have Table 5.16 – this latter is an example of multiple "phantom" zeros and so does not converge.

i	x_i	Y_i
1	-0.3000	-0.2474
2	0.3000	0.4998
3	0.2850	0.0425
4	0.2063	0.0032
5	0.2000	0.0000

Table 5.12: Design generated by criterion C_2 for function $f_1(\cdot)$ with $\sigma_0^2 = \sigma_b^2 = \theta = 1$ and $\mu_0 = 0$, using the extended model.

i	x_i	Y_i
1	-0.3000	-0.2474
2	0.3000	0.4998
3	0.1992	-0.0004
4	0.2000	0.0000

Table 5.13: "Inverse interpolation" design for function $f_1(\cdot)$ with $\sigma_0^2 = \sigma_b^2 = \theta = 1$ and $\mu_0 = 0$, using the extended model.

i	x_i	Y_i
1	-0.3000	0.0193
2	0.3000	0.0198
3	-0.3000	0.0193
4	-0.3375	0.0187
5	-0.4989	0.0004
6	-0.5002	-0.0001
7	-0.1604	0.0198
8	-0.5000	0.0000

Table 5.14: Design generated by criterion C_1 for function $f_2(\cdot)$ with $\sigma_0^2 = \sigma_b^2 = \theta = 1$ and $\mu_0 = 0$, using the extended model.

i	x_i	Y_i
1	-0.3000	0.0193
2	0.3000	0.0198
3	-0.3001	0.0193
4	-0.3424	0.0186
5	-0.5026	-0.0008
6	-0.4996	0.0001
7	-0.5000	0.0000

Table 5.15: Design generated by criterion C_2 for function $f_2(\cdot)$ with $\sigma_0^2 = \sigma_b^2 = \theta = 1$ and $\mu_0 = 0$, using the extended model.

i	x_i	Y_i
1	-0.3000	0.0193
2	0.3000	0.0198
3	-3.0000	-360287
4	-0.3000	0.0193
5	-0.2993	0.0193
6	-0.2989	0.0193
7	-0.3122	0.0191
8	-0.3198	0.0190
9	-0.3401	0.0186
10	-0.3505	0.0183
11	-0.3827	0.0171
12	-0.4244	0.0141
13	-0.2655	0.0196
14	-0.2104	0.0197
15	-0.4487	0.0112

Table 5.16: "Inverse interpolation" design for function $f_2(\cdot)$ with $\sigma_0^2 = \sigma_b^2 = \theta = 1$ and $\mu_0 = 0$, using the extended model.

5.6 Other models

So far we have just considered one basic correlation function $\rho(x - x_*) = e^{-\theta(x-x_*)^2}$, but we could use the other correlation functions from Chapter 2 to repeat the computations for either model. The approximations to the function are still continuous and differentiable everywhere except at the design points. To try and compare these different models, it would be useful to match "like with like" – the θ parameter in each class of models is different. In $\rho_1(\cdot)$ and $\rho_2(\cdot)$, the smaller the value of θ the higher the correlation between observations, but in the other two the opposite is true. One such way of comparing these parameters is to look at the "average" correlation between any two slopes (as this is zero) we will look at the correlation integrated over all separations. We see for $\rho_2(\cdot)$ this integral is $\sqrt{(\pi/\theta)}$; for $\rho_1(\cdot)$, $2/\theta$; for $\rho_{l+}(\cdot)$, θ; and for $\rho_{c+}(\cdot)$, $\frac{3}{4}\theta$. For the θ parameters we examined in Section 5.2 we have the equivalences tabulated in Table 5.17. We can use these equivalences to examine the

θ in $\rho_2(\cdot)$	θ in $\rho_1(\cdot)$	θ in $\rho_{l+}(\cdot)$	θ in $\rho_{c+}(\cdot)$
0.1	0.36	5.60	7.47
1.0	1.13	1.77	2.36
10.0	3.57	0.56	0.75

Table 5.17: Table of equivalent θs for alternative "slope" correlation functions

correlation and covariance surfaces for these in a similar way to those for the case of $\rho_2(\cdot)$, see Figures 5.16 to 5.24, and traces of functions that fit the model, see Figures 5.25 to 5.27 for various values of θ. We see that the use of different correlation structures do not make much difference to the correlation structure, especially the difference between $\rho_2(\cdot)$ and $\rho_{c+}(\cdot)$. It appears that the question of which correlation function should be used, might only be of importance in the computational side of the matrix inversion – $\rho_{c+}(\cdot)$ and $\rho_{l+}(\cdot)$ have advantages here as the elements of C for different xs sufficiently far apart are 0 and not very small. We can also compare designs and predictors produced for the different covariance structures but, as you can guess from the similarities in the correlation functions, these are little different

from the designs produced in the case of $\rho_2(\cdot)$

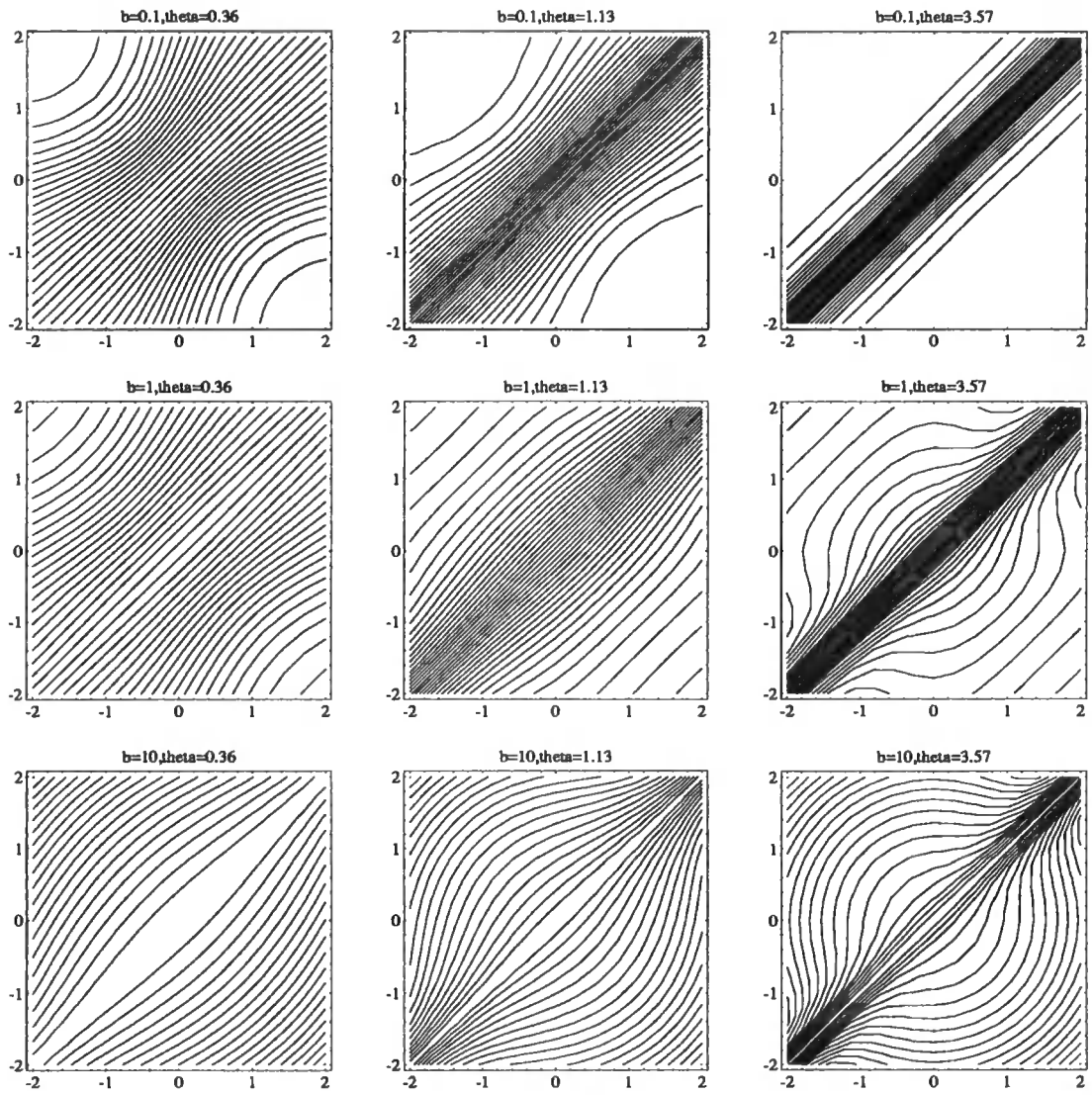

Figure 5.16: Correlation between $Y(x)$ and $Y(x^*)$, using $\rho_1(d)$ as the slope correlation

Other models as in equation 5.4 have been investigated, but have not been found to be satisfactory, with the correlation matrices either being singular, or having to make additional assumptions about the parameters. An example of one such model assumes that the $G(\cdot)$ process is stationary, with covariance $\sigma_g^2\rho_2(\cdot)$. If we write

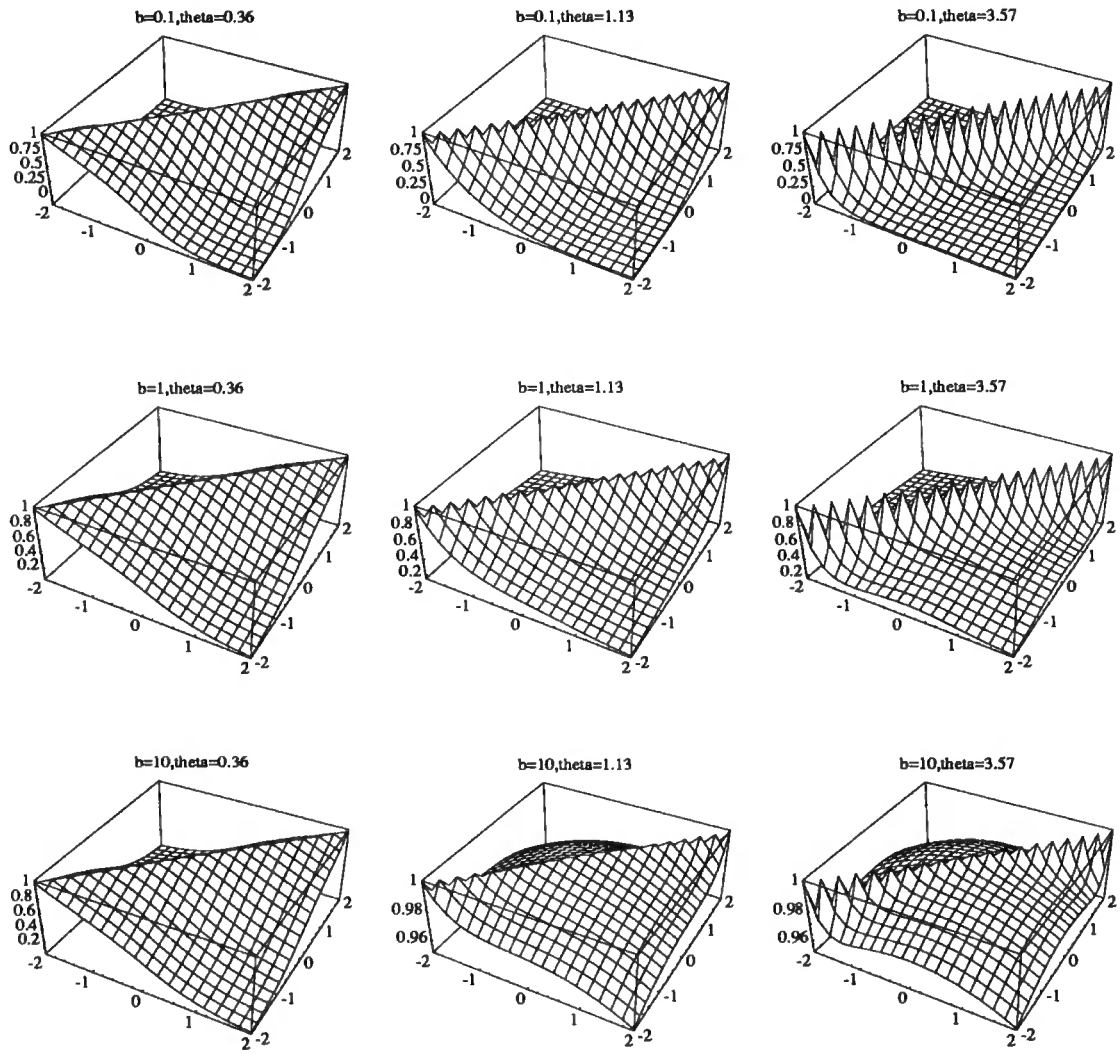

Figure 5.17: Correlation between $Y(x)$ and $Y(x^*)$, using $\rho_1(d)$ as the slope correlation

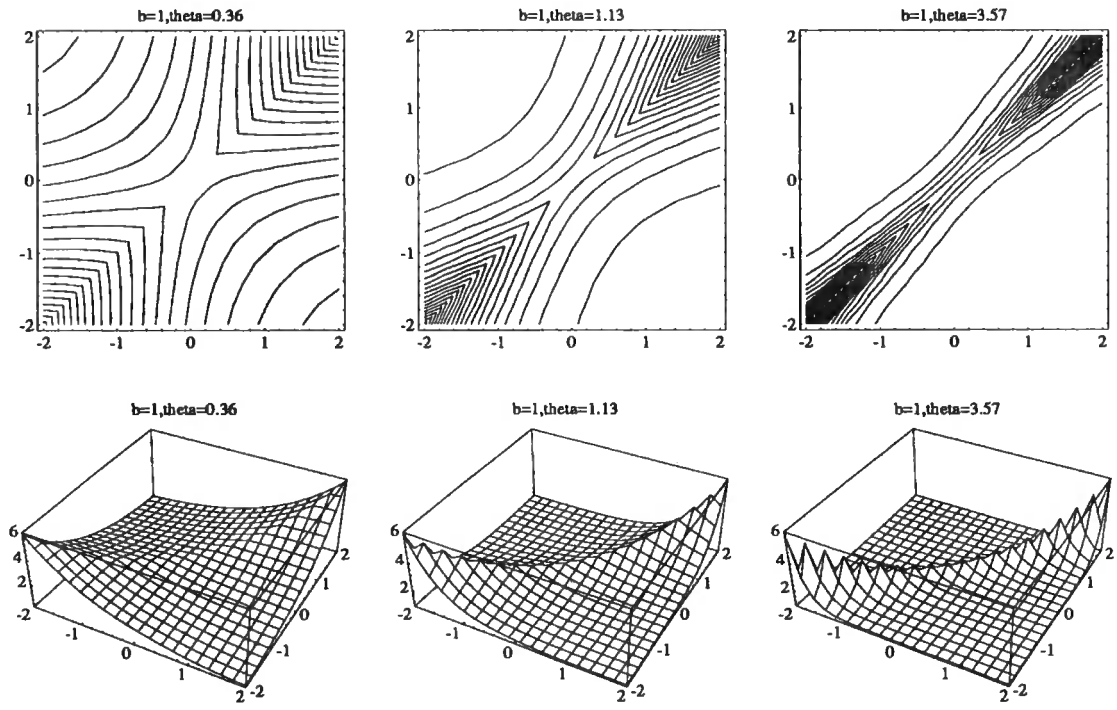

Figure 5.18: Covariance between $Y(x)$ and $Y(x^*)$, using $\rho_1(d)$ as the slope correlation

$\theta = \frac{1}{2\tau^2}$ we have,

$$\begin{aligned} \mathrm{P}(Y(x)) &= \mathrm{P}(G(x)) - \mathrm{P}(G(X_0)) = 0 && (5.46) \\ \mathrm{Cov}_{X_0}(Y(x), Y(x_*)) &= \mathrm{Cov}_{X_0}(G(x), G(x_*)) - \mathrm{Cov}_{X_0}(G(x), G(X_0)) \\ &\quad - \mathrm{Cov}_{X_0}(G(x_*), G(X_0)) + \mathrm{Var}_{X_0}(G(X_0)) \\ &= \sigma_g^2 \left(1 + e^{-\frac{1}{2\tau^2}(x-x_*)^2} - e^{-\frac{1}{2\tau^2}(x-X_0)^2} + e^{-\frac{1}{2\tau^2}(x_*-X_0)^2}\right) && (5.47) \end{aligned}$$

From a purely linear Bayes viewpoint we can do no more, but if we assume additionally, that X_0 is distributed normally with mean μ_0 and variance σ_0^2, we can take expectations giving us

$$\begin{aligned} \mathrm{Cov}(Y(x), Y(x_*)) &= \mathrm{P}(\mathrm{Cov}_{X_0}(Y(x), Y(x_*))) \\ &= \sigma_g^2 \left(1 + e^{-\frac{1}{2\tau^2}(x-x_*)^2} - \frac{e^{-\frac{(x-\mu_0)^2}{2(\tau^2+\sigma_0^2)}} + e^{-\frac{(x_*-\mu_0)^2}{2(\tau^2+\sigma_0^2)}}}{\sqrt{1+\sigma_0^2/\tau^2}}\right) && (5.48) \end{aligned}$$

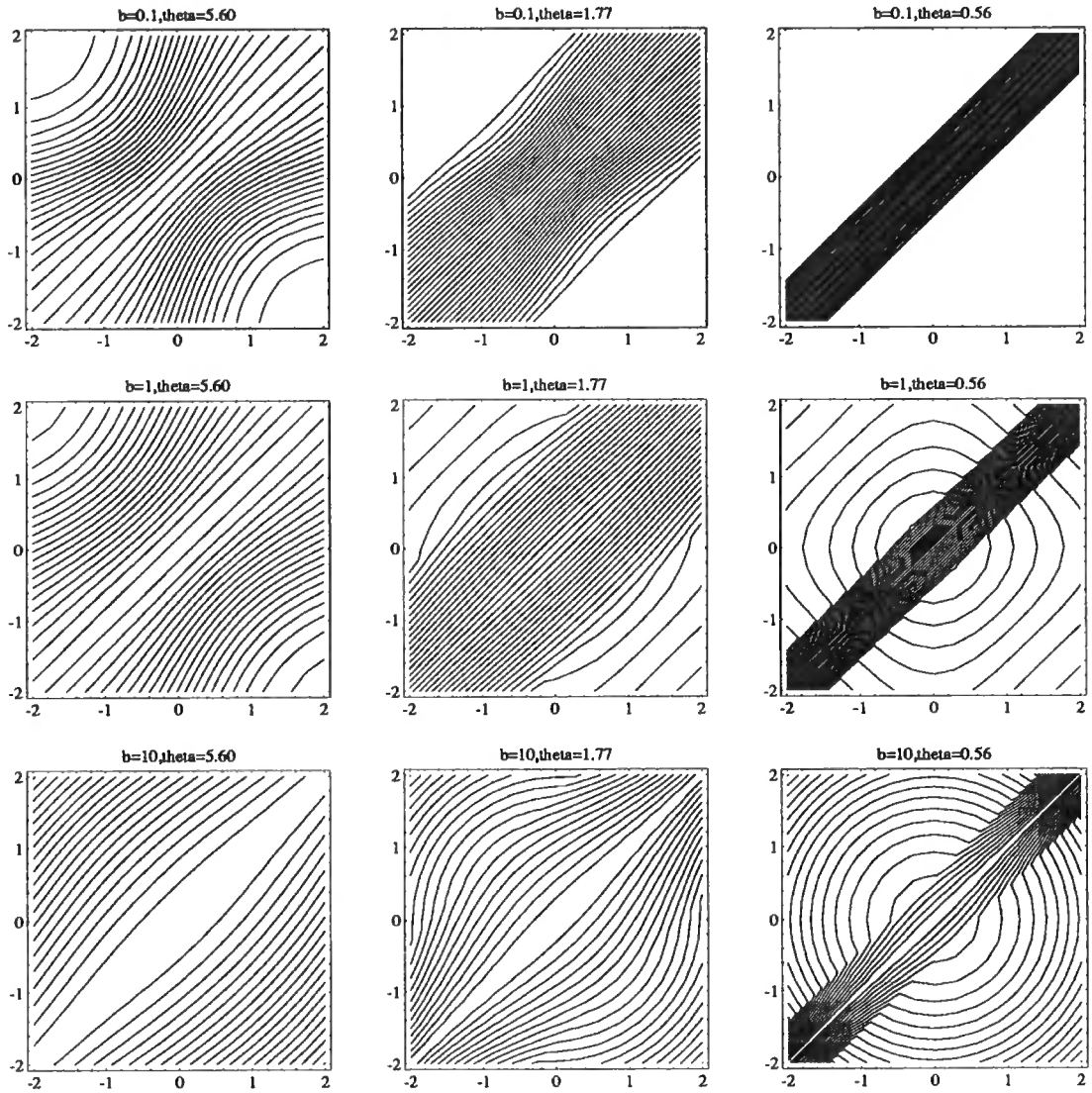

Figure 5.19: Correlation between $Y(x)$ and $Y(x^*)$, using $\rho_{l+}(d)$ as the slope correlation

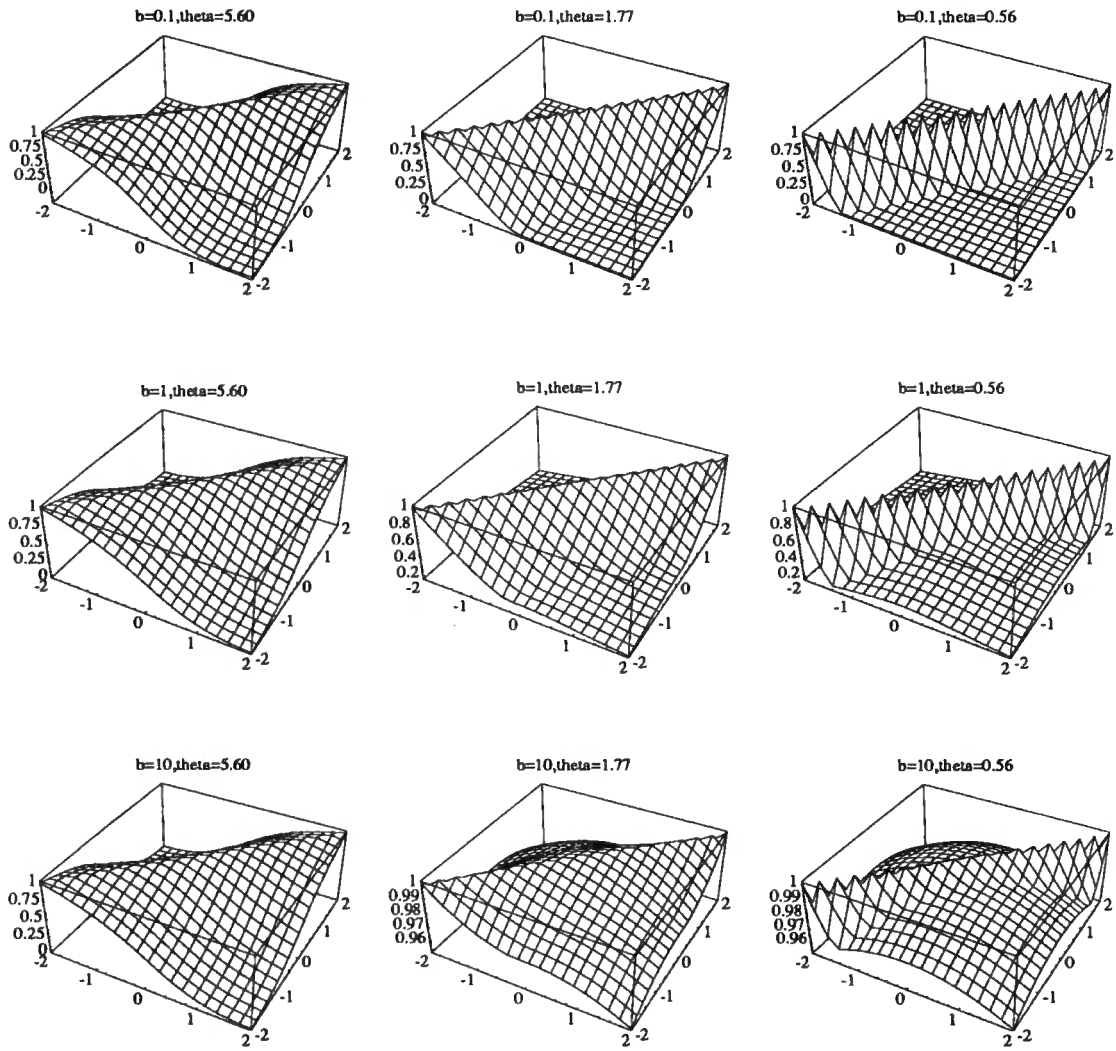

Figure 5.20: Correlation between $Y(x)$ and $Y(x^*)$, using $\rho_{l+}(d)$ as the slope correlation

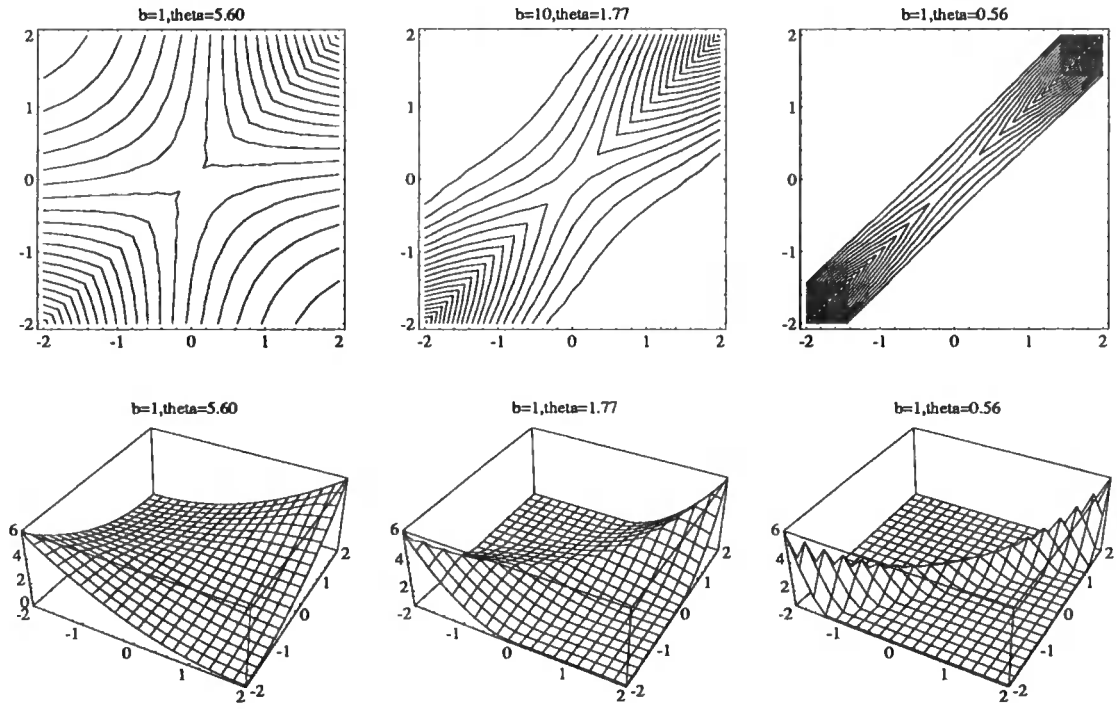

Figure 5.21: Covariance between $Y(x)$ and $Y(x^*)$, using $\rho_{l+}(d)$ as the slope correlation

which can be used in a similar way to the correlation structures we obtained earlier, and using either one of the variance modified criteria, or the "Newtonian" estimates for X_0, as again we have no linear estimate of X_0 available.

5.7 Eliciting prior beliefs

One of the hardest parts of belief analysis is the elicitation of prior beliefs, although in the Bayes Linear context this is usually much easier than for a full Bayes specification. In the problem we are considering we need to elicit information about the zero and the slope process. The first order previsions are not too difficult to elicit, but the second order structure can be. This is the reason for explicitly removing any initial dependence between $B(x)$ and X_0, as variances are generally easier for people to estimate than covariances or correlations.

One way of obtaining the variance parameters is to ask the scientist to lay down

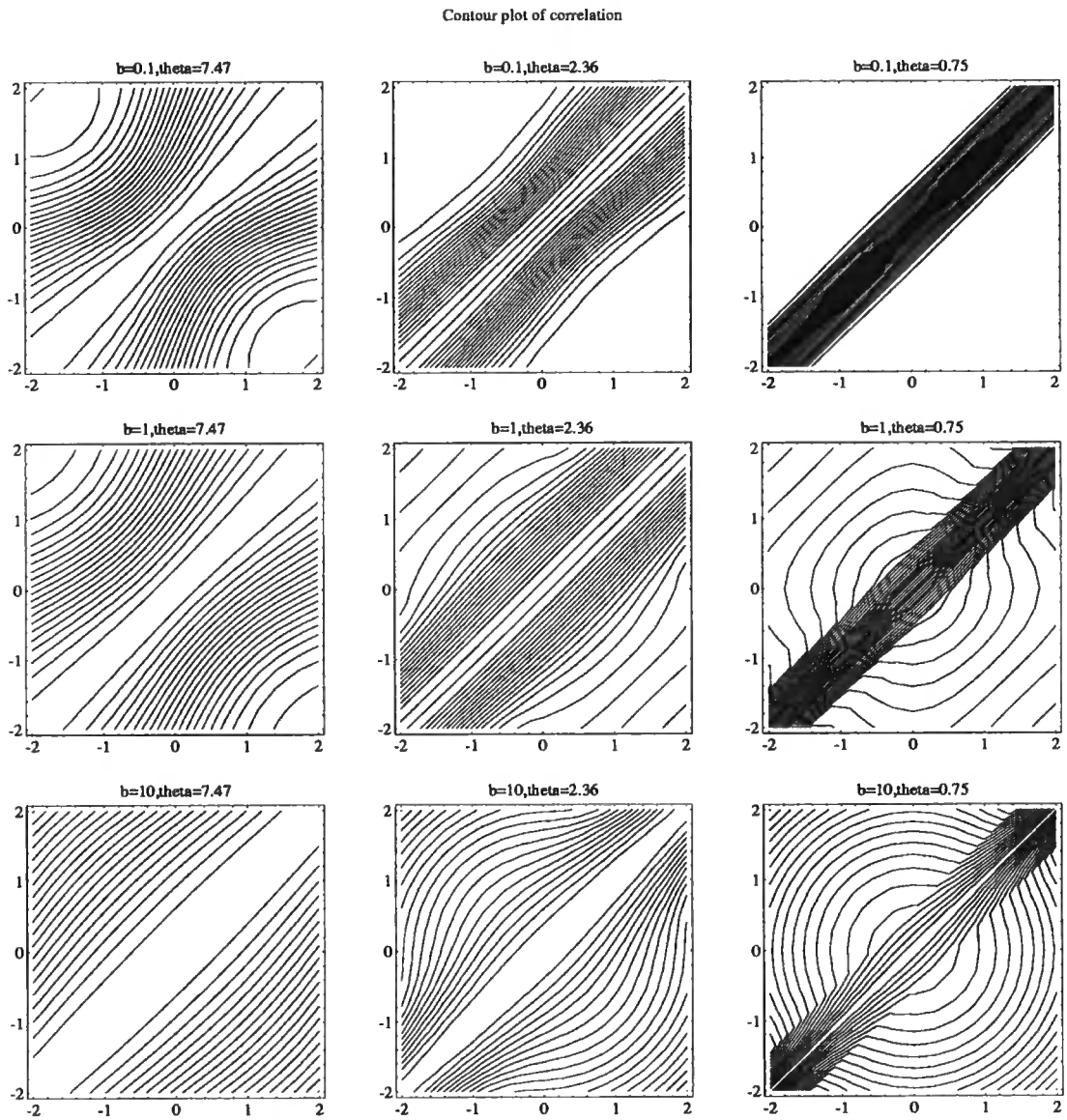

Figure 5.22: Correlation between $Y(x)$ and $Y(x^*)$, using $\rho_{c+}(d)$ as the slope correlation

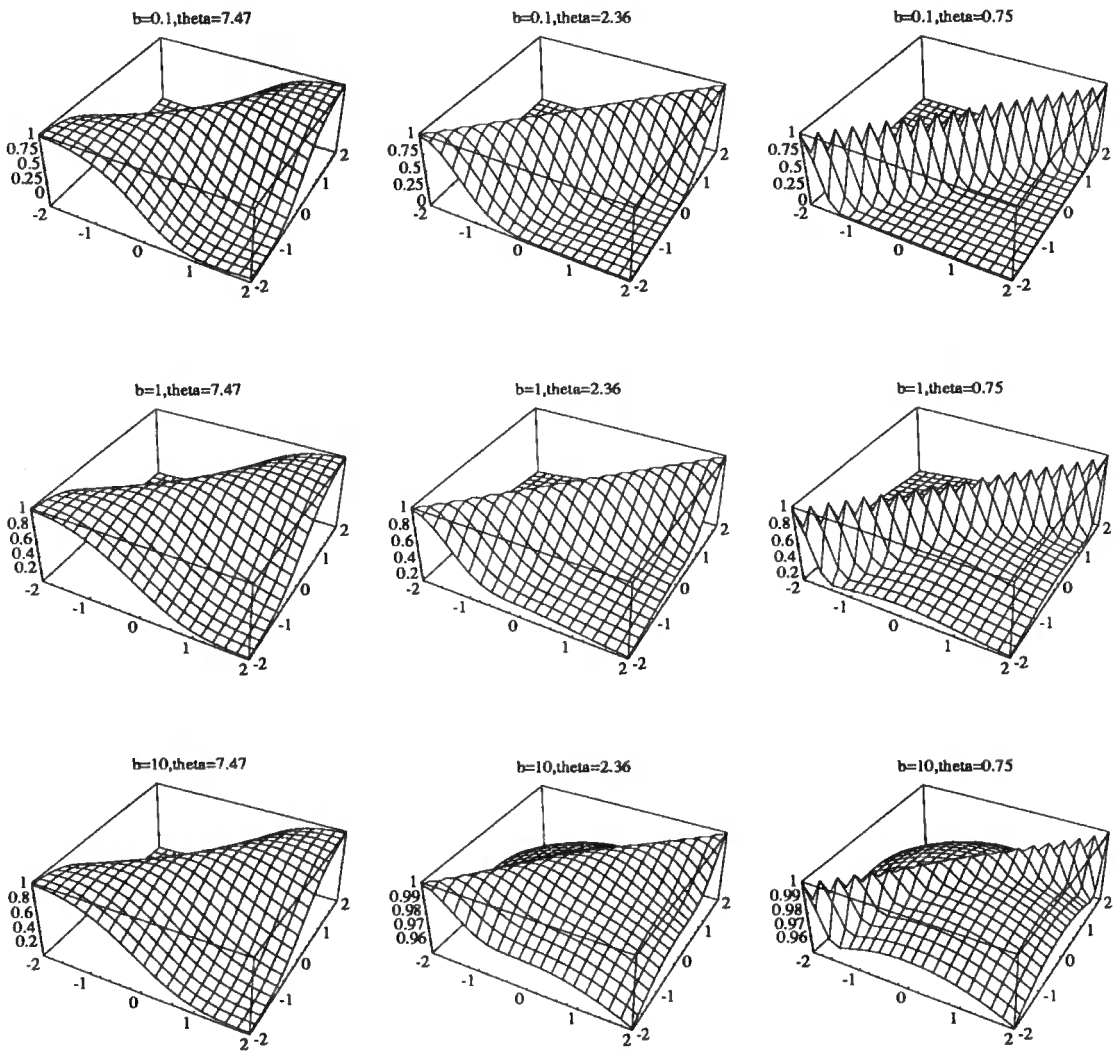

Figure 5.23: Correlation between $Y(x)$ and $Y(x^*)$, using $\rho_{c+}(d)$ as the slope correlation

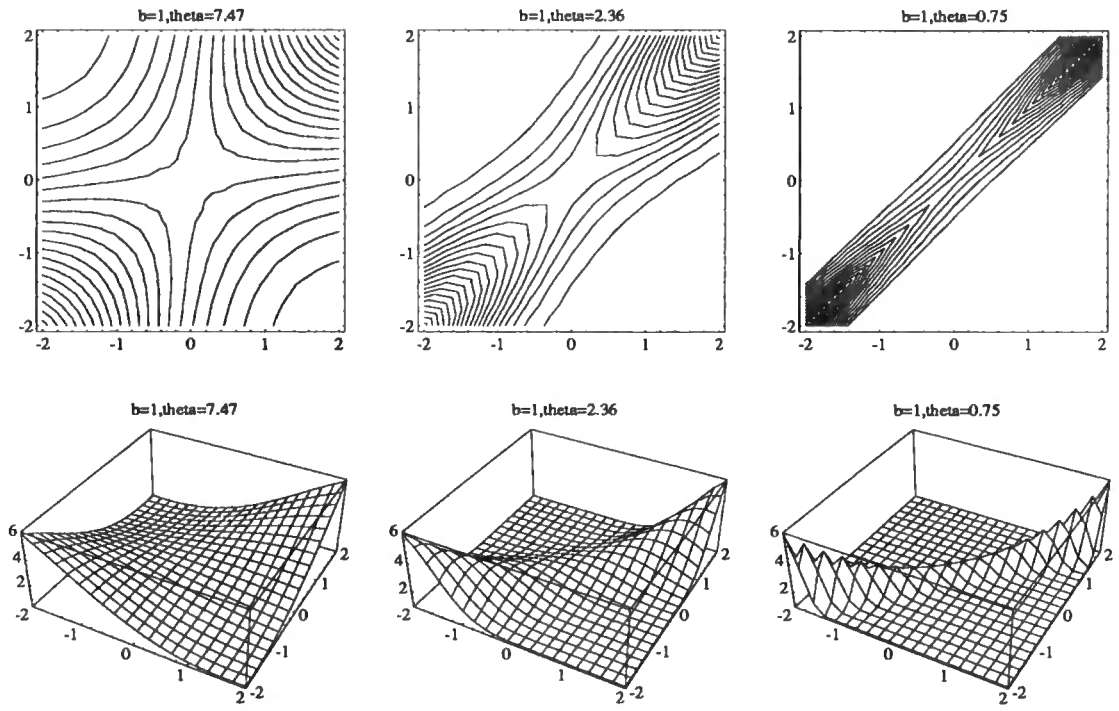

Figure 5.24: Covariance between $Y(x)$ and $Y(x^*)$, using $\rho_{c+}(d)$ as the slope correlation

intervals around the mean in which he expects the parameter to line in with a given percentage. For example if he specifies intervals of 70% or 95%, these give ideas of one or two standard deviations from the mean.

Finally we have to choose an initial value for the smoothness parameter θ. Three possible ways of doing this are

We can use the results of Section 5.9, in which we see that $\text{Var}(B'(x)) = 2\theta\text{Var}(B(x))$, and $\text{Var}(Y'(x)) = \sigma_b^2 + 2\theta(\text{Var}(Y(x)) - b^2\sigma_0^2)$. So θ measures the amount of variability in the first derivative relative to the function itself.

The second is linked to the first, the variability of the first derivative is related to the second derivative, as the variability of the function is related to the first derivative. Therefore θ is a measure of the magnitude of the second derivative to the first, if the magnitude of the second derivative is large in comparison to that of the first then θ is large, and if the magnitude of it is small, then θ is small.

The third method looks at the relationship between two slopes at different values

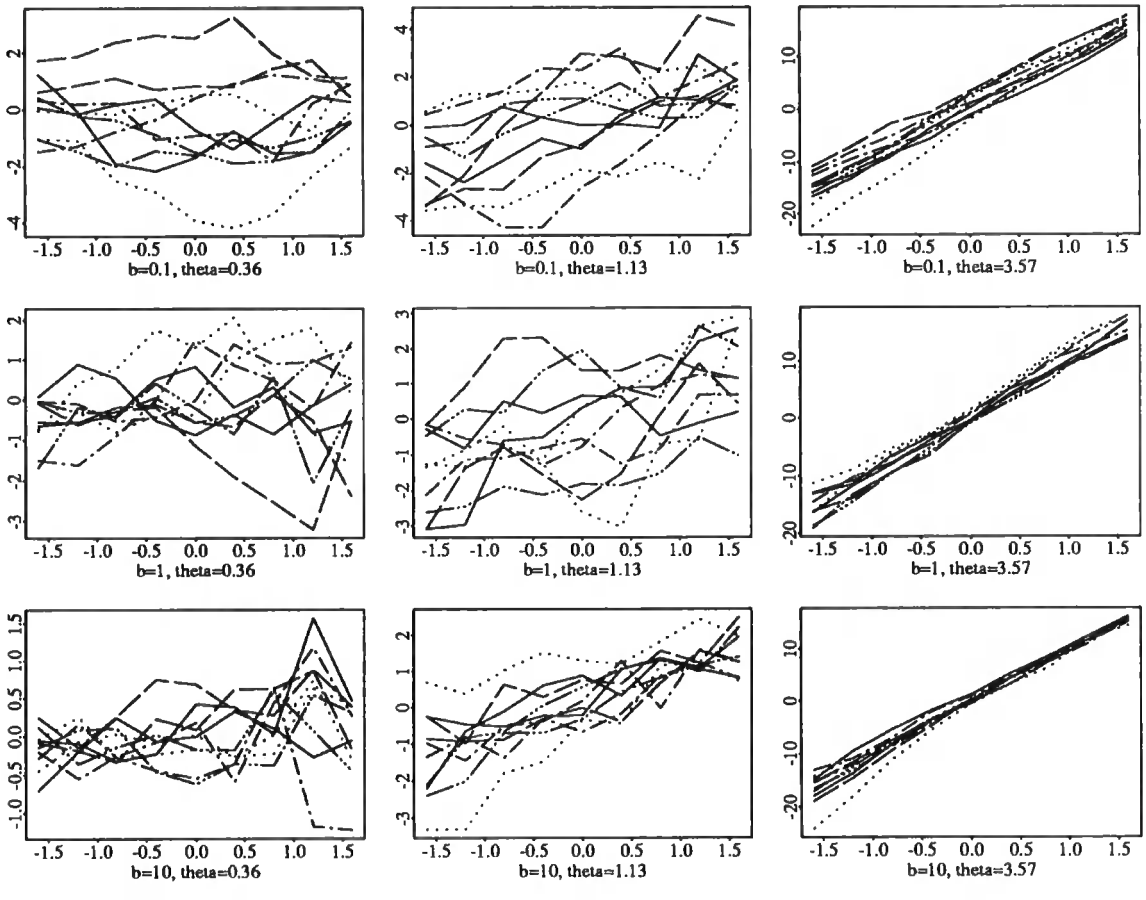

Figure 5.25: Traces of $Y(x)$, using $\rho_1(d)$ as the slope correlation

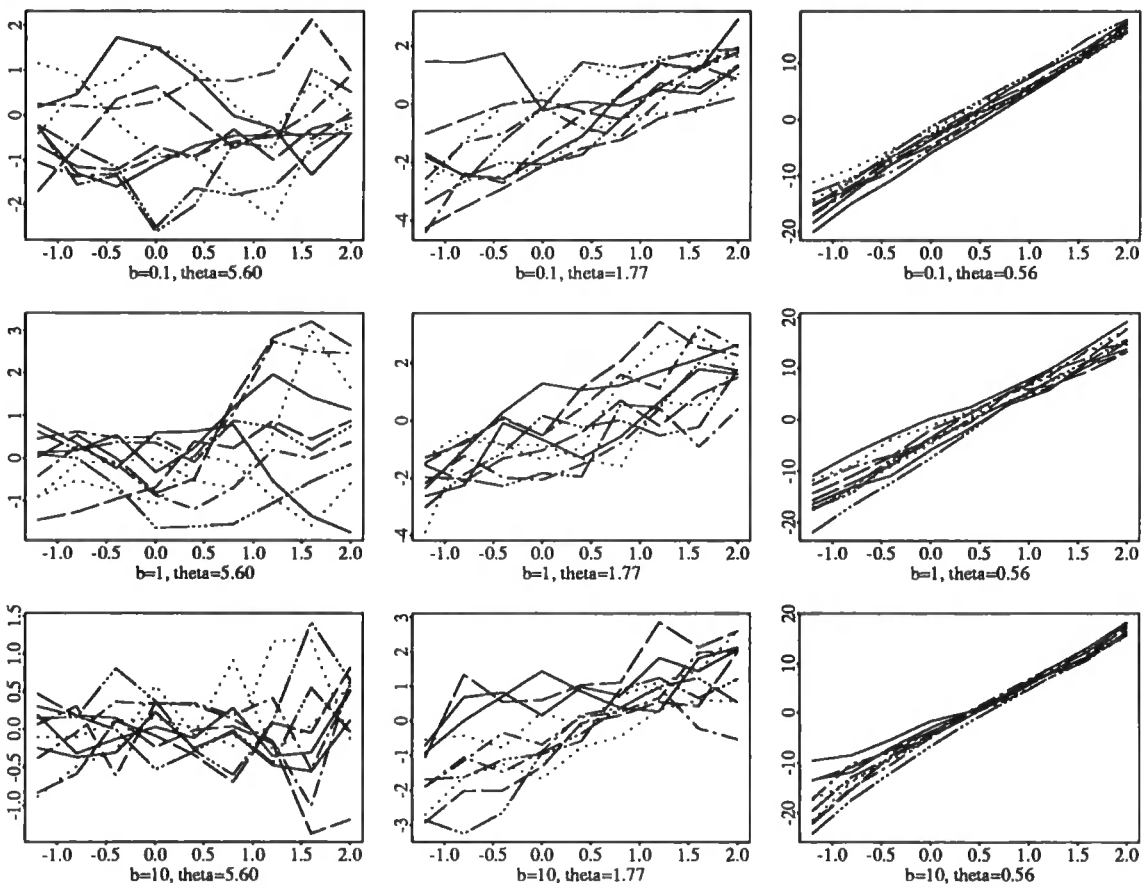

Figure 5.26: Traces of $Y(x)$, using $\rho_{l+}(d)$ as the slope correlation

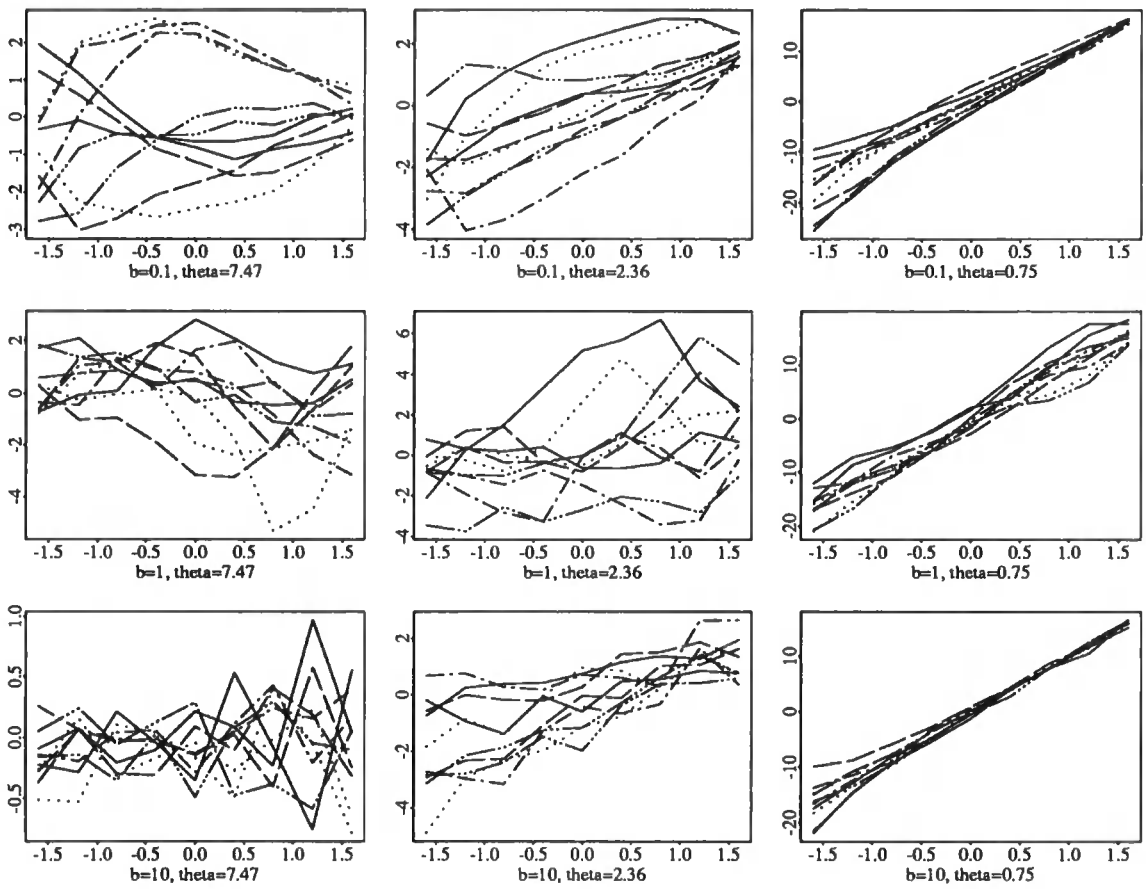

Figure 5.27: Traces of $Y(x)$, using $\rho_{c+}(d)$ as the slope correlation

of x. Here we can do this by asking the scientist to sketch regions on a graph where he thinks that two values of B for different xs will be, as in Figure 5.28, and then working out the correlation from the diagram. Assuming the variance of $B(\cdot)$ is constant, it can easily be shown that $\rho(x - x^*)$ is given by $\frac{U-V}{U+V}$, so that$\theta = \frac{\log(U+V)-\log(U-V)}{(x-x^*)^2}$.

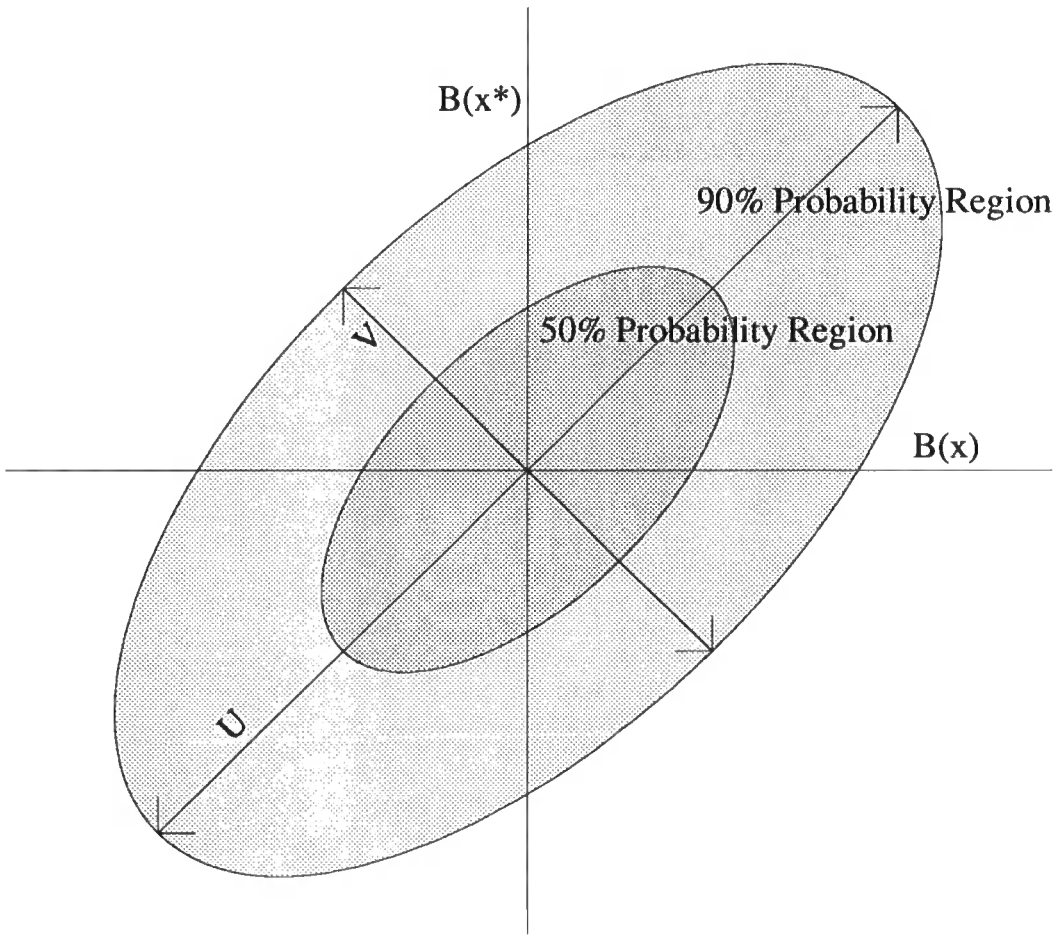

Figure 5.28: Estimating correlation from "probability regions" : $\rho = \frac{U-V}{U+V}$

5.8 A model with random error

So far we have been considering the interpolation problem, where the approximation function fits the observed data exactly at the design points. If however we assume the function is no longer deterministic we can add an extra "error" term.

$$Y(x_i) = B(x_i)(x_i - X_0) + E_i \tag{5.49}$$

This is easy to implement in a Linear Bayes framework. All we need to do is alter the second-order structure. We can however use this approach even when the simulation is truly deterministic. We assume we evaluate the function with a small error (with variance σ_ϵ^2) – which in general will be the case, for example if we use a differential equation solver there will always be some numerical error. This now gives us a non-interpolatory approximation, which numerical analysts know as *smoothing*. This has one major advantage over the exact methods in the previous section, the variance matrix R is replaced by $R + \sigma_\epsilon^2 I$. This reduces the condition number of the variance matrix, making it easier to invert, and making the approximant more stable when there is a large number of observations.

The model in equation (5.49) can be extended further to have non-independent errors, for example, we might assume the error process is stationary, with a similar correlation structure to that of the $B(\cdot)$'s, but with a much larger smoothness parameter $\eta \gg \theta$, i.e.

$$\mathrm{Cov}(E(x), E(x_*)) = \sigma_\epsilon^2 e^{-\eta(x-x_*)^2} \tag{5.50}$$

We can consider this new term as a systematic departure of the computer model from the "real-world" phenomena it is modelling as in Blight and Ott[1975], see Section 2.6. This is also similar to Schagen's approach of long and short range trends also reviewed in Chapter 2.

5.9 Including the derivative

We may have other information about the function as well as its actual value. For example, if the problem we are solving involves differential equations we may also obtain information about the derivative of the function at the design points. We

would like to include this additional information into our methods.

Assuming we keep the same model as before, we can get this information, by taking derivatives of the correlation function, because as both prevision and differentiation are bounded linear operators we can interchange them. So we have

$$
\begin{aligned}
Y'(x) &= B(x) + (x - X_0)B'(x) \\
\mathrm{P}(B'(x)) &= 0 \\
\mathrm{P}(Y'(x)) &= b \\
\mathrm{Cov}(B'(x), B(x^*)) &= -2\theta(x - x^*)\sigma_b^2 e^{-\theta(x-x^*)^2} \\
\mathrm{Cov}(B'(x), B'(x^*)) &= -2\theta(2\theta(x - x^*)^2 - 1)\sigma_b^2 e^{-\theta(x-x^*)^2} \\
\mathrm{Cov}(Y(x), B'(x^*)) &= 2\theta(x - \mu_0)(x - x^*)e^{-\theta(x-x^*)^2} \\
\mathrm{Cov}(Y'(x), X_0) &= 0 \\
\mathrm{Cov}(Y'(x), B(x^*)) &= (1 - 2\theta(x - \mu_0)(x - x^*))e^{-\theta(x-x^*)^2} \\
\mathrm{Cov}(Y'(x), B'(x^*)) &= 2\theta[(x - x^*) - (x - \mu_0)(1 - 2\theta(x - x^*)^2)]\sigma_b^2 e^{-\theta(x-x^*)^2} \\
\mathrm{Cov}(Y'(x), Y(x^*)) &= [(x^* - \mu_0) - 2\theta(x - x^*)(\sigma_0^2 + (x - \mu_0)(x^* - \mu_0))]\sigma_b^2 e^{-\theta(x-x^*)^2} \\
\mathrm{Cov}(Y'(x), Y'(x^*)) &= (1 - 2\theta(x - x^*)^2)[1 + 2\theta(\sigma_0^2 + (x - \mu_0)(x^* - \mu_0))]\sigma_b^2 e^{-\theta(x-x^*)^2}
\end{aligned}
$$

We note that in these equations θ appears outside the exponential term, and so, as we mentioned in Section5.7, we can use this to estimate θ.

If the we take function $(Y_1, \ldots, Y_n)$ and derivative $(Y_1', \ldots, Y_n')$ evaluations at $x_1, \ldots, x_n$ and defining

$$
\mathrm{Var}\begin{pmatrix} \boldsymbol{Y} \\ \boldsymbol{Y}' \end{pmatrix} = \begin{pmatrix} C_{YY} & C_{YY'} \\ C_{Y'Y} & C_{Y'Y'} \end{pmatrix}
$$

It should be noted that we do not necessarily have to have pairs of function and derivative evaluations, but can use function evaluations and derivatives for different values of x.

We can get adjusted previsions and covariances for random quantities that are correlated to the function or its derivative. In general if we were to adjust G by $Y_1, \ldots, Y_n$ and $Y_1', \ldots, Y_n'$, where our prior beliefs are

$$\begin{aligned} \mathrm{P}(G) &= g_0 \\ \mathrm{Var}(G) &= \sigma_g^2 \\ \mathrm{Cov}(G, \boldsymbol{Y}) &= \boldsymbol{c}^{\mathrm{T}} \\ \mathrm{Cov}(G, \boldsymbol{Y}') &= \boldsymbol{c}'^{\mathrm{T}} \end{aligned}$$

we get

$$\begin{aligned} \mathrm{P}_{\boldsymbol{Y}\boldsymbol{Y}'}(g) &= g_0 + \boldsymbol{c}^{\mathrm{T}} C_{YY}^{-1}(\boldsymbol{Y} - b\boldsymbol{x} + b\mu_0\mathbf{1}) + (\boldsymbol{c}' - C_{Y'Y}C_{YY}^{-1}\boldsymbol{c})^{\mathrm{T}} \qquad (5.51) \\ &\quad (C_{Y'Y'} - C_{Y'Y}C_{YY}^{-1}C_{YY'})^{-1}(\boldsymbol{Y}' - b\mathbf{1} - C_{Y'Y}C_{YY}^{-1}(\boldsymbol{Y} - b\boldsymbol{x} + b\mu_0\mathbf{1})) \\ \mathrm{Var}([g/\boldsymbol{Y}\boldsymbol{Y}']) &= \sigma_g^2 - \boldsymbol{c}^{\mathrm{T}} C_{YY}^{-1}\boldsymbol{c} - \qquad (5.52) \\ &\quad (\boldsymbol{c}' - C_{Y'Y}C_{YY}^{-1}\boldsymbol{c})^{\mathrm{T}}(C_{Y'Y'} - C_{Y'Y}C_{YY}^{-1}C_{YY'})^{-1}(\boldsymbol{c}' - C_{Y'Y}C_{YY}^{-1}\boldsymbol{c}) \end{aligned}$$

So we can use this extra information to gain knowledge about the function and its zero. It can be seen that this extra information is in the form of an adjustment to the adjustments made by the function evaluations only.

If we use our simple example $f_1(\cdot)$, with our standard beliefs $\sigma_0^2 = \sigma_b^2 = b = \theta = 1$ and $\mu_0 = 0$, to test this predictor, and for a design criteria choose our next point to be the location of the zero of the interpolant, we find the zero in just three function/derivative evaluation pairs $x_1 = 0$, $x_2 = 0.190236$ and $x_3 = 0.200000$. We can easily see the extra information we receive from the data by comparing the graphs of the error bounds, which are at least two orders of magnitude smaller near the design points than in the case of function evaluations only. See Figures 5.29 and 5.30.

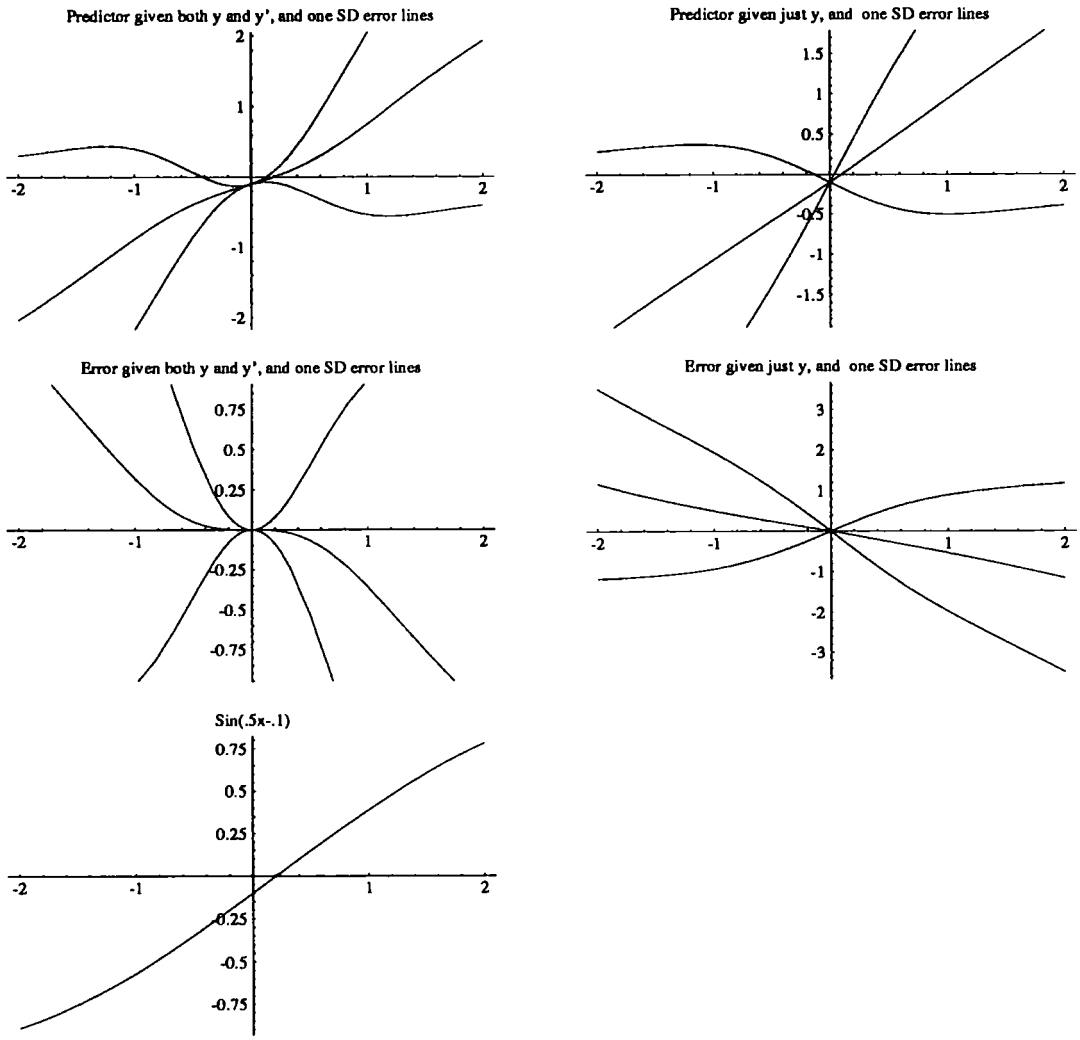

Figure 5.29: Plots of $f_1(\cdot)$, and its approximants, given one observation at 0, with model parameters $\theta = b = \sigma_0^2 = \sigma_b^2 = 1$ and $\mu = 0$

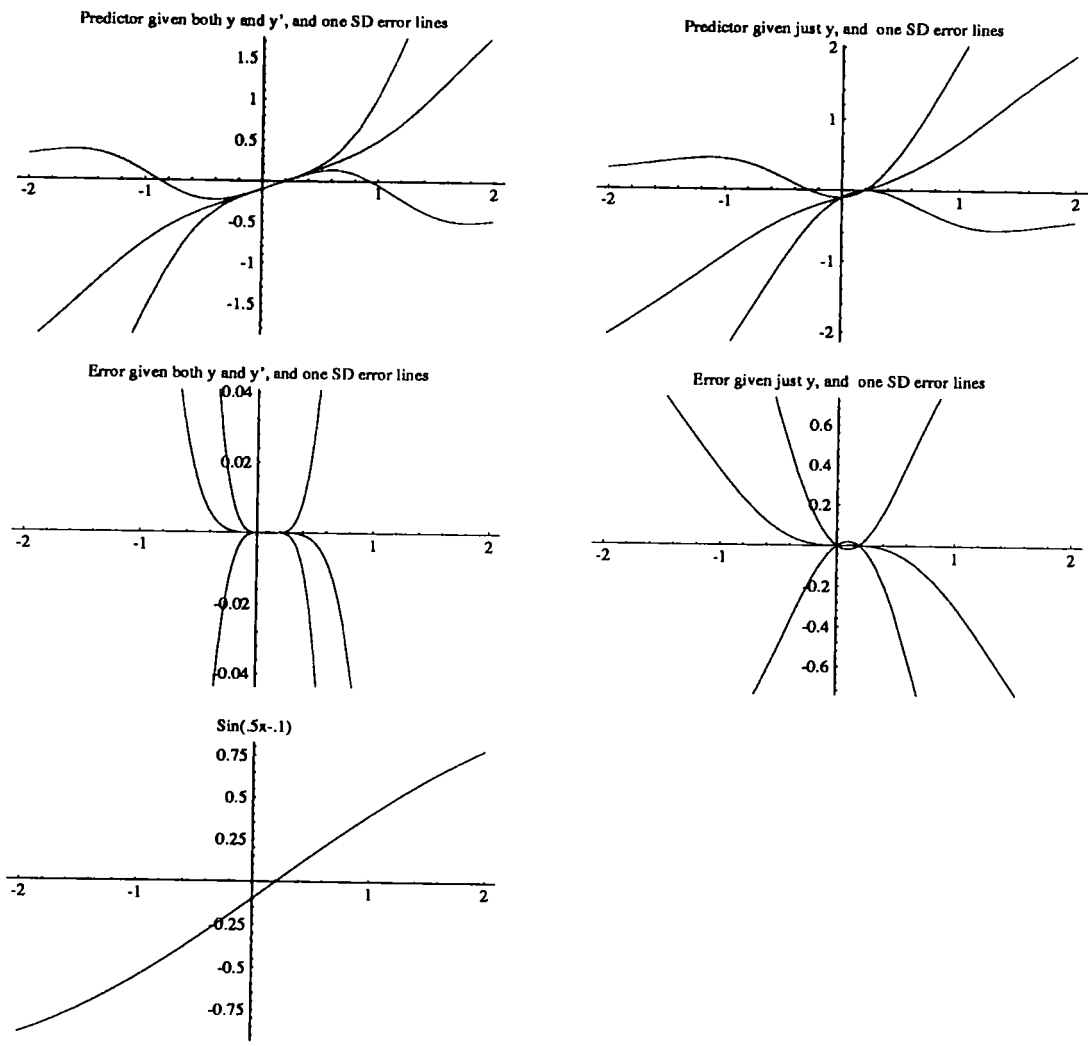

Figure 5.30: Plots of $f_1(\cdot)$, and its approximants, given two observation at 0 and 0.190236, with model parameters $\theta = b = \sigma_0^2 = \sigma_b^2 = 1$ and $\mu = 0$

5.10 The multi-variate response problem

We now consider the case when each observation $\boldsymbol{Y}(x)$ is a m-vector of responses, we can still use a variant of the approach we used when it was univariate. Here we modify the model as follows,

$$\boldsymbol{Y}(x) = \boldsymbol{B}(x)(x - X_0) \tag{5.53}$$

We can now write down our beliefs about the $\boldsymbol{B}(\cdot)$s. For any particular $B_i(\cdot)$ we have similar beliefs to those in the univariate model, i.e.

$$\begin{aligned} \mathrm{P}(B_i(x)) &= b_i & (5.54)\\ \mathrm{Cov}(B_i(x), B_i(x_*)) &= \sigma_i^2 \rho(x - x_*) & (5.55) \end{aligned}$$

We also model our beliefs between two parts of the response for a given x,

$$\mathrm{Cov}(B_i(x), B_j(x)) = \sigma_i \sigma_j r_{ij} \tag{5.56}$$

Finally the correlation between two parts of two separate responses is the product of these, i.e.

$$\mathrm{Cov}(B_i(x), B_j(x_*)) = \sigma_i \sigma_j r_{ij} \rho(x - x_*) \tag{5.57}$$

In many applications, the vector of responses $\boldsymbol{Y}(\cdot)$ can be thought of as a finite collection of observations from a continuous time series $\boldsymbol{Y} = (Y(t_1), \ldots, Y(t_m))$. Therefore, the equations can be re-written as

$$\begin{aligned} \mathrm{P}(B(x,t)) &= b(t) & (5.58)\\ \mathrm{Cov}(B(x,t), B(x_*,t)) &= \sigma(t)^2 \rho(x - x_*) & (5.59)\\ \mathrm{Cov}(B(x,t), B(x,t_*)) &= \sigma(t)\sigma(t_*) r(t - t_*) & (5.60)\\ \mathrm{Cov}(B(x,t), B(x_*,t_*)) &= \sigma(t)\sigma(t_*) r(t - t_*) \rho(x - x_*) & (5.61) \end{aligned}$$

We now derive our implied beliefs about $\boldsymbol{Y}(\cdot)$ from our beliefs about X_0 and $\boldsymbol{B}(\cdot)$ as we did for the univariate response. This gives us:
(in the case when we are considering discrete observations)

$$\begin{aligned}
\mathrm{P}(Y_i(x_*)) &= b_i(x-\mu_0) && (5.62)\\
\mathrm{Cov}(X_0, Y_i(x)) &= -\sigma_0^2 b_i && (5.63)\\
\mathrm{Cov}(B_i(x), Y_j(x_*)) &= \sigma_i\sigma_j r_{ij}\rho(x-x_*)(x_*-\mu_0) && (5.64)\\
\mathrm{Cov}(Y_i(x), Y_j(x_*)) &= \sigma_i\sigma_j r_{ij}\rho(x-x_*)(\sigma_0^2+(x-\mu_0)(x_*-\mu_0)) + \sigma_0^2 b_i b_j && (5.65)
\end{aligned}$$

(in the case when we are considering observations from a continuous time series.)

$$\begin{aligned}
\mathrm{P}(Y(x,t)) &= b(t)(x-\mu_0) && (5.66)\\
\mathrm{Cov}(X_0, Y(x,t)) &= -\sigma_0^2 b(t) && (5.67)\\
\mathrm{Cov}(B(x,t), Y(x_*,t_*)) &= \sigma(t)\sigma(t_*)r(t-t_*)\rho(x-x_*)(x_*-\mu_0) && (5.68)\\
\mathrm{Cov}(Y(x,t), Y(x_*,t_*)) &= \sigma(t)\sigma(t_*)r(t-t_*)\rho(x-x_*)(\sigma_0^2+(x-\mu_0)(x_*-\mu_0)) \\
&\quad +\sigma_0^2 b(t)b(t_*) && (5.69)
\end{aligned}$$

It should be noted that all these equations consist of terms of the form $f(x,x_*)\times g(t,t_*)$ or $f(x,x_*)\times g_{ij}$, and so the matrix equations have a Krönecker product form. This seems natural, as we are using a grid and there is an "independency" between the x and the t; and desireable, as Krönecker product matrices have nice properties which we would like to be able to use.

If we observe $\boldsymbol{Y}_1,\ldots,\boldsymbol{Y}_n$ at $x_1,\ldots,x_n$, then we can derive posterior estimates for quantities of interest similar to those in the univariate case. First we will need to

define some additional notation.

$$\begin{aligned}
\boldsymbol{B}_i &= (B_1(x_i),\ldots,B_m(x_i)) && \text{in the discrete case}\\
&= (B(x_i,t_1),\ldots,B(x_i,t_m)) && \text{in the continuous case}\\
\boldsymbol{b} &= (b_1,\ldots,b_m)^{\mathrm{T}} && \text{in the discrete case}\\
&= (b(t_1),\ldots,b(t_m)) && \text{in the continuous case}\\
\Sigma_{ij} &= \sigma_i\sigma_i r_{ij} && \text{in the discrete case}\\
&= \sigma(t_i)\sigma(t_j)r(t_i-t_j) && \text{in the continuous case}\\
\boldsymbol{Y} &= (\boldsymbol{Y}_1^{\mathrm{T}},\ldots,\boldsymbol{Y}_n^{\mathrm{T}})^{\mathrm{T}}
\end{aligned} \tag{5.70}$$

So $\Sigma = \mathrm{Var}(\boldsymbol{B}_i)$. Our Bayes linear estimates for X_0, $\boldsymbol{B}(\cdot)$ and $\boldsymbol{Y}(\cdot)$ are

$$\mathrm{P}_{\boldsymbol{Y}}(X_0) = \frac{\mu_0-\sigma_0^2(\mathbf{1}^{\mathrm{T}}C^{-1}\otimes\boldsymbol{b}^{\mathrm{T}}\Sigma^{-1})\boldsymbol{Y}+\sigma_0^2\mathbf{1}^{\mathrm{T}}C^{-1}\boldsymbol{x}\boldsymbol{b}^{\mathrm{T}}\Sigma^{-1}\boldsymbol{b}}{1+\sigma_0^2\mathbf{1}^{\mathrm{T}}C^{-1}\mathbf{1}\boldsymbol{b}^{\mathrm{T}}\Sigma^{-1}\boldsymbol{b}} \tag{5.71}$$

$$\mathrm{P}_{\boldsymbol{Y}}(\boldsymbol{B}(x_*)) = (1-\boldsymbol{d}^{\mathrm{T}}C^{-1}[\boldsymbol{x}-\mathrm{P}_{\boldsymbol{Y}}(X_0)\mathbf{1}])\boldsymbol{b}+(\boldsymbol{d}^{\mathrm{T}}C^{-1}\otimes I)\boldsymbol{Y} \tag{5.72}$$

$$\mathrm{P}_{\boldsymbol{Y}}(\boldsymbol{Y}(x_*)) = (\boldsymbol{c}^{\mathrm{T}}C^{-1}\otimes I)\boldsymbol{Y}+([x_*-\mathrm{P}_{\boldsymbol{Y}}(X_0)]-\boldsymbol{c}^{\mathrm{T}}C^{-1}[\boldsymbol{x}-\mathrm{P}_{\boldsymbol{Y}}(X_0)\mathbf{1}])\boldsymbol{b} \tag{5.73}$$

where all notation is the same as in Section 5.2. The adjusted variances and covariances are

$$\mathrm{Var}[X_0/\boldsymbol{Y}] = \frac{\sigma_0^2}{1+\sigma_0^2\mathbf{1}^{\mathrm{T}}C^{-1}\mathbf{1}\boldsymbol{b}^{\mathrm{T}}\Sigma^{-1}\boldsymbol{b}} \tag{5.74}$$

$$\begin{aligned}
\mathrm{Cov}([\boldsymbol{B}(x_*)/\boldsymbol{Y}],[\boldsymbol{B}(\tilde{x}_*)/\boldsymbol{Y}]) &= \rho(x_*-\tilde{x}_*)\Sigma-\boldsymbol{d}^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{d}}\Sigma+\\
&\quad \boldsymbol{d}^{\mathrm{T}}C^{-1}\mathbf{1}\mathbf{1}^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{d}}\mathrm{Var}[X_0/\boldsymbol{Y}]\boldsymbol{b}\boldsymbol{b}^{\mathrm{T}}
\end{aligned} \tag{5.75}$$

$$\begin{aligned}
\mathrm{Cov}([\boldsymbol{B}(x_*)/\boldsymbol{Y}],[\boldsymbol{Y}(\tilde{x}_*)/\boldsymbol{Y}]) &= (\tilde{x}_*-\mu_0)\rho(x_*-\tilde{x}_*)\Sigma-\boldsymbol{d}^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{c}}\Sigma+\\
&\quad \boldsymbol{d}^{\mathrm{T}}C^{-1}\mathbf{1}(\mathbf{1}^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{c}}-1)\mathrm{Var}[X_0/\boldsymbol{Y}]\boldsymbol{b}\boldsymbol{b}^{\mathrm{T}}
\end{aligned} \tag{5.76}$$

$$\begin{aligned}
\mathrm{Cov}([\boldsymbol{Y}(x_*)/\boldsymbol{Y}],[\boldsymbol{Y}(\tilde{x}_*)/\boldsymbol{Y}]) &= c(x_*,\tilde{x}_*)\rho(x_*-\tilde{x}_*)\Sigma-\boldsymbol{c}^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{c}}\Sigma+\\
&\quad (\boldsymbol{c}^{\mathrm{T}}C^{-1}\mathbf{1}-1)(\mathbf{1}^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{c}}-1)\mathrm{Var}[X_0/\boldsymbol{Y}]\boldsymbol{b}\boldsymbol{b}^{\mathrm{T}}
\end{aligned} \tag{5.77}$$

In the continuous case we can also predict the outcome for any t as we have the correlation between $Y(t)$ and $Y(t_*)$ for all t and t_*.

These results are analogous to the univariate case, and can be used in a similar way. We note that the adjusted covariances above are all linear combinations of Σ and $\mathrm{Var}[X_0/\boldsymbol{Y}]\boldsymbol{b}\boldsymbol{b}^{\mathrm{T}}$

5.11 The full problem

The model can be extended further to consider the case when $\boldsymbol{x}$ is also multivariate (a $q \times 1$ vector). We build our model in a similar way to before. We again modify the model of the response $\boldsymbol{Y}(\cdot)$ to allow for this, and we think of it as a sum of processes in each direction. We will consider only the case of discrete observations from a time series as the notation is easier to follow

$$Y(\boldsymbol{x},t) = \sum_{i=1}^{q} B_i(\boldsymbol{x},t)(x_i - X_{0i}) \tag{5.78}$$

where $B_i(\cdot,\cdot)$ the slope process in the direction of x_i. We assume that these $B_i(\cdot,\cdot)$ processes are independent, and that each one has a similar structure to those in the previous section, i.e.

$$\mathrm{P}(B_i(\boldsymbol{x},t)) = b_i(t) \tag{5.79}$$

$$\mathrm{Cov}(B_i(\boldsymbol{x},t), B_j(\boldsymbol{x}_*,t_*)) = \delta_{ij}\sigma_i(t)\sigma_i(t_*)\rho_i(\boldsymbol{x}-\boldsymbol{x}_*)r_i(t-t_*) \tag{5.80}$$

We complete the model by assuming that $\boldsymbol{X}_0$ has prevision $\boldsymbol{\mu}_0$ and variance Σ_0. We can again calculate our implied beliefs about the response.

$$\mathrm{P}(Y(\boldsymbol{x},t)) = \sum_{i=1}^{q} b_i(t)(x_i - \mu_{0i}) \tag{5.81}$$

$$\begin{aligned}\mathrm{Cov}(Y(\boldsymbol{x},t), Y(\boldsymbol{x}^*,t^*)) &= \boldsymbol{b}(t)^{\mathrm{T}}\Sigma_0\boldsymbol{b}(t^*) + \\ &\sum_{i=1}^{q}\sigma_i(t)\sigma_i(t_*)\rho_i(\boldsymbol{x}-\boldsymbol{x}^*)r_i(t-t^*)[(x_i-\mu_{0i})(x_i^*-\mu_{0i}) + \Sigma_{0ii}]\end{aligned} \tag{5.82}$$

Given that our design is $\{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n\} \otimes \{t_1,\ldots,t_m\}$, and we have observation

$\{\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n\}$, where $\boldsymbol{Y}_i = (Y(\boldsymbol{x}_i, t_1), \ldots, Y(\boldsymbol{x}_i, t_m))^{\mathrm{T}}$, we have

$$\mathrm{P}(\boldsymbol{Y}_i) = B(\boldsymbol{x}_i - \boldsymbol{\mu}_0) \tag{5.83}$$

$$\mathrm{Cov}(\boldsymbol{Y}_i, \boldsymbol{Y}_j) = B\Sigma_0 B^{\mathrm{T}} + \sum_{k=1}^{q} \Sigma_k \rho_k(\boldsymbol{x}_i - \boldsymbol{x}_j)[(x_{ik} - \mu_{0k})(x_{jk} - \mu_{0k}) + \Sigma_{0kk}] \tag{5.84}$$

where

$$\begin{aligned} \boldsymbol{b}_i &= (b_i(t_1), \ldots, b_i(t_m))^{\mathrm{T}} \\ \Sigma_k &= (\sigma_k(t)\sigma_k(t_*)r_k(t_i - t_j))_{1 \le i,j \le m} \\ B &= (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_q) \end{aligned}$$

Putting $\boldsymbol{x} = (\boldsymbol{x}_1^{\mathrm{T}}, \ldots, \boldsymbol{x}_n^{\mathrm{T}})^{\mathrm{T}}$, $\boldsymbol{Y} = (\boldsymbol{Y}_1^{\mathrm{T}}, \ldots, \boldsymbol{Y}_n^{\mathrm{T}})^{\mathrm{T}}$ and $C_k = (\rho_k(\boldsymbol{x}_i - \boldsymbol{x}_j)[(x_{ik} - \mu_{0k})(x_{jk} - \mu_{0k}) + \Sigma_{0kk}])_{1 \le i,j \le n}$ we have,

$$\mathrm{P}(\boldsymbol{Y}) = (B \otimes I)(\boldsymbol{x} - \boldsymbol{\mu}_0 \otimes \mathbf{1}) \tag{5.85}$$

$$\mathrm{Var}(\boldsymbol{Y}) = (B \otimes \mathbf{1})(\Sigma_0 \otimes I)(B \otimes \mathbf{1})^{\mathrm{T}} + \sum_{k=1}^{q} \Sigma_k \otimes C_k \tag{5.86}$$

It should be noted that the *nice* algebraic results of the previous section where the covariance matrices factor into Krönecker products no longer hold. Therefore no *nice* algebraic results can be obtained – unless we assume that either the Σ_k or C_k are the same for all k – we have to work with the matrix as a whole unit, and cannot factorise it into x and t parts. The numerical problems associated with this grow in size as the matrix gets larger, i.e, as the dimension $\boldsymbol{Y}(\cdot)$ and the number of observations increases, this leads us onto the next section.

5.12 Computational difficulties

As has been repeatedly mentioned, the covariance matrices that arise in these problems are often ill-conditioned; i.e., if we try to solve $A\boldsymbol{x} = \boldsymbol{y}$, the maximum error in $\boldsymbol{x}$ is very large in comparison to the error in $\boldsymbol{y}$. This can lead to unstable solutions, in

fact quite often the error can be many orders of magnitude larger, than their values. These are most apparent when computed variances are very large and negative!

It was noted in Section 5.8 that if we add a random error term to the problem, then the condition number decreases. We can use this device (adding a very small error term to reduce the condition number) even when we are not considering models with random error; compare this with ridge regression in normal error analyses. We do not need to restrict ourselves to independent errors, but can use very short range correlated errors, as in Schagen[1980a]. In many cases the actual computer model does not match the actual physical world precisely, so we can consider this additional error term as a systematic departure of the computer model (either in the stage of modelling nature or implementing it on the computer) from nature, as in Blight and Ott[1975].

5.13 The full Bayes model

For completeness we include a full Bayes model for the simplest case, when $Y(\cdot)$ is a univariate function of a single variable x.

We extend our second order beliefs to full prior distribution assumptions by adding that $B(\cdot)$ has a multivariate normal distribution, and X_0 has a normal distribution, and we now assume that $B(x)$ and X_0 are independent, not just uncorrelated.

Then

$$\pi_{X0}(x_0) = \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{1}{2\sigma_0^2}(x_0-\mu_0)^2} \tag{5.87}$$

$$\pi_B(\boldsymbol{b}) = \frac{1}{\sqrt{(2\pi\sigma_b^2)^n |R|}} e^{-\frac{1}{2\sigma_b^2}(\boldsymbol{b}-b\mathbf{1})^T R^{-1}(\boldsymbol{b}-b\mathbf{1})} \tag{5.88}$$

$$l(\boldsymbol{y}|\boldsymbol{b}, x_0) = \prod_{i=1}^{n} \delta(y_i - b_i(x_i - x_0)) \tag{5.89}$$

where $R_{ij} = \rho(|x_i - x_j|)$ and $\delta(\cdot)$ is the Dirac-delta function. We can combine (5.88)

and (5.89) together to obtain the likelihood of $\boldsymbol{y}$ given x_0,

$$l(\boldsymbol{y}|x_0) = \frac{1}{\sqrt{(2\pi\sigma_b^2)^n|R|}} e^{-\frac{1}{2\sigma_b^2}(\boldsymbol{z}-b\mathbf{1})^T R^{-1}(\boldsymbol{z}-b\mathbf{1})}, \tag{5.90}$$

where $z_i = \frac{y_i}{x_i - x_0}$.

We now calculate our posterior distributions X_0 given n and $n+1$ observations, the $n+1^{st}$ being at x^*,

$$\pi_n(x_0|\boldsymbol{y}) \propto e^{-\frac{1}{2\sigma_0^2}(x_0-\mu_0)^2 - \frac{1}{2\sigma_b^2}(\boldsymbol{z}-b\mathbf{1})^T R^{-1}(\boldsymbol{z}-b\mathbf{1})} \tag{5.91}$$

$$\pi_{n+1}(x_0|\boldsymbol{y}.y(x^*)) \propto \pi_n(x_0|\boldsymbol{y}) e^{-\frac{1}{2\sigma_b^2}\frac{(z^* - b - \boldsymbol{r}^{*\mathrm{T}}R^{-1}(\boldsymbol{z}-b\mathbf{1}))^2}{1-\boldsymbol{r}^{*\mathrm{T}}R^{-1}\boldsymbol{r}^*}} \tag{5.92}$$

where $z^* = \frac{y(x^*)}{x^* - x_0}$, and $r_i^* = \rho(|x_i - x^*|)$. Thus, to update the relative posterior density at any point is relatively easily, needing only to solve one set of linear equations for each x^*, $R\boldsymbol{s}^* = \boldsymbol{r}^*$. When we add a new design point x^* to the grid, we note that the inverse of the new covariance matrix R^{*-1} can also be easily updated using $\boldsymbol{s}^*$,

$$R^{*-1} = \begin{pmatrix} R^{-1} + k\boldsymbol{s}^*\boldsymbol{s}^{*\mathrm{T}} & -k\boldsymbol{s}^* \\ -k\boldsymbol{s}^{*\mathrm{T}} & k \end{pmatrix}$$

where $k = (1 - \boldsymbol{s}^{*\mathrm{T}}\boldsymbol{r}^*)^{-1}$.

We can also compute the predictive distribution for $Y(x^*)$ given the previous observations.

$$p(y(x^*)|x_0, \boldsymbol{y}) = \frac{1}{\sqrt{(2\pi\sigma_b^2)(1-\boldsymbol{r}^{*\mathrm{T}}R^{-1}\boldsymbol{r}^*)}} e^{-\frac{1}{2\sigma_b^2}\frac{(z^* - b - \boldsymbol{r}^{*\mathrm{T}}R^{-1}(\boldsymbol{z}-b\mathbf{1}))^2}{1-\boldsymbol{r}^{*\mathrm{T}}R^{-1}\boldsymbol{r}^*}}$$

then integrating this with respect to $\pi_n(x_0|\boldsymbol{y})$ gives us

$$p(y(x^*)|\boldsymbol{y})$$

$$= \frac{\int_{-\infty}^{+\infty} dx_0 e^{\frac{1}{2\sigma_0^2}(x_0-\mu_0)^2-\frac{1}{2\sigma_b^2}(\boldsymbol{z}-b\mathbf{1})^T R^{-1}(\boldsymbol{z}-b\mathbf{1})-\frac{1}{2\sigma_b^2}\frac{(z^*-b-\boldsymbol{r}^{*T}R^{-1}(\boldsymbol{z}-b\mathbf{1}))^2}{1-\boldsymbol{r}^{*T}R^{-1}\boldsymbol{r}^*}}}{\sqrt{(2\pi\sigma_b^2)(1-\boldsymbol{r}^{*T}R^{-1}\boldsymbol{r}^*)}\int_{-\infty}^{+\infty} dx_0 e^{\frac{1}{2\sigma_0^2}(x_0-\mu_0)^2-\frac{1}{2\sigma_b^2}(\boldsymbol{z}-b\mathbf{1})^T R^{-1}(\boldsymbol{z}-b\mathbf{1})}} \tag{5.93}$$

These posterior distributions are not analytically integrable, even if $\rho(\cdot,\cdot)$ has a "nice" form, and so have to be done computationally. The posterior distribution for X_0 is multi-modal, with zeros at the design points, often climbing very steeply (if one of the design points has just missed a zero), so the numerical integration requires many ordinates in these regions to obtain accurate predictions. Even the now popular Gibbs sampler can not be used in this example without modification, as it can not cope with distributions where there are zeros (especially in this example as the zero regions can be very wide and flat), as it can get trapped in one of the spikes with no chance of escape.

To avoid the problems of integration we can use the mode of the posterior distribution of x_0 as our new estimate. We will use this method with our first example $f_1(\cdot)$, with the model parameters $\mu_0 = 0$ and $\sigma_b^2 = \sigma_0^2 = b = \theta = 1$, at each stage adding the mode of the distribution, see figure 5.31. The design points used were 0.000000, 0.098876, 0.206302, 0.200128 and 0.200000 (The last one is the zero we are looking for). Once we have taken four function evaluations the posterior distribution of x_0 around 0.2 becomes very spiked (as you can see the plotting program failed to find it after three points), with the spike width being less then 10^{-6}. This means we have to be very careful when we are searching for the mode that we have not missed out the spike. If we can work out the posterior distribution exactly (up to a scale factor) this should not pose any problem, as the posterior is unimodal between design points, and we can use the "Golden-ratio" search algorithm. However, if we have even a very small numerical error in the posterior calculations then this fluctuation will cause the above methods to fail.

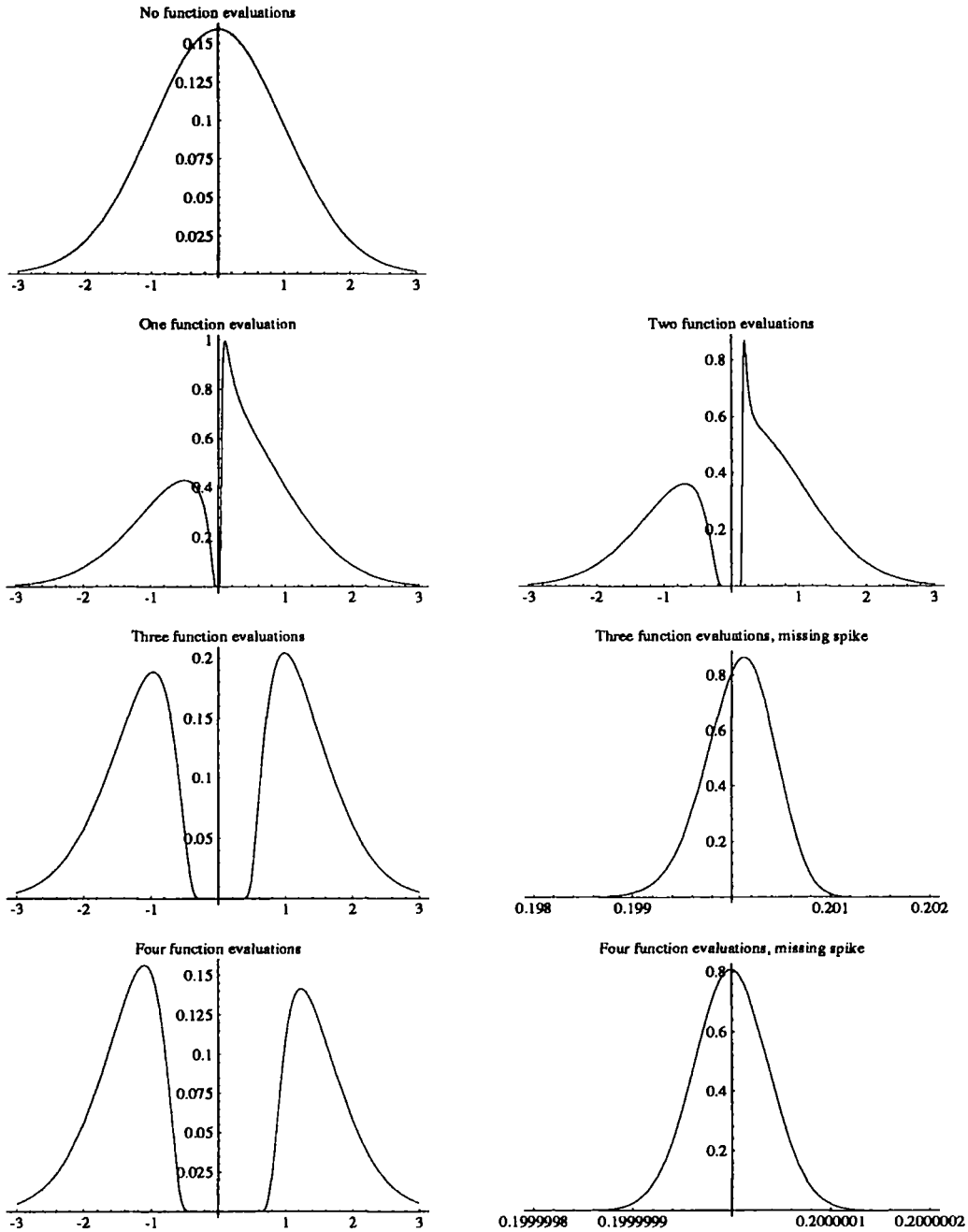

Figure 5.31: (Un-normalised) posterior probability densities of X_0, after each point has been added, (after 3 and 4 points the spike near the zero is included separately, as is too sharp for the graph plotter to find)

This method is nearly impossible to peform, even for small designs, as the computational problems far out-weigh any advantages that might be achieved by this method.

5.14 Conclusion

In this chapter we have developed various models and associated criteria for locating the zero of a deterministic function. We have found three methods which appear to work well, the "Inverse Interpolation" method, the variance modified methods, and the "Newtonian" estimates. Although the first two converge the quickest in most examples they do have a more complicated design criteria to choose the next point (either the solution of a non-linear equation or the optimization of a function), whereas the "Newtonian" method is simpler. These two different types of criteria can therefore be used for different jobs: the first for more expensive functions – where the added work in choosing would not be two costly; and the second for relatively cheap functions – where the added costs out-weigh the benefits.

Chapter 6

Conclusions and Further Avenues of Research

We have seen throughout this document how we can use statistical methods to develop numerical procedures to solve deterministic problems. In Chapter 2 we saw that sometimes this has produced tried and tested numerical procedures, for example linear interpolation and the "Mid-point rule" from the Brownian motion prior in Section 2.2.2, although new designs for different criteria were produced. However, in general they produce completely new methods. One such example is derived from the use of stationary stochastic processes to model the function (see Sacks *et al*[1989a, 1989b], Schagen[1979, 1980a, 1980b, 1984], O'Hagan[1978, 1990] and Currin *et al*[1991]). In light of this in Chapter 4, we used the Bayes linear methods of Goldstein[1981, 1986, 1987, 1988b, 1991] (reviewed in Chapter 3) to arrive at a criterion for producing designs for this model which, with some simple algebra, greatly simplified the calculations required to find optimal designs, especially in higher dimensions when a lattice design region and a factorisable correlation structure was chosen.

In relation to our problem, we have tried to develop new (Bayesian) methods to find the zero of a deterministic function, mainly trying to use the Bayes linear methodology. Due to computational difficulties we have only be able to fully explore

these methods in the case of a single valued function of a single variable, which is a much simpler problem than the full problem. We developed a statistical model of the function which explicitly includes our beliefs about the location of the zero. We then used this statistical model to make predictions about aspects of the real function. To locate the zero, we developed different search strategies, using various estimators of the zero of the function: the Bayes linear estimate $\mathrm{P}_{\mathbf{Y}}(X_0)$; the zero of the predictor; the minimum of the squared expectation of the predictor; a "Newtonian" estimator, based on our revised beliefs about the slope of the function; and the mode of the posterior distribution in a full Bayes analysis. Of these by far and away the poorest was the naive linear estimator, the rest performing much better. The "Newtonian" estimates were just behind the others in convergence speed (in number of observations), but we had an explicit equation for the location of the zero. So we need to consider the ease of obtaining the next point in the design, with the "Newtonian" we required no optimization or zero finding, whereas with the others much time had to be spent finding the maxima or the zero. This problem is amplified when we increase the number of dimensions and design points . In the first case we can no longer rely on the criterion being unimodal in certain intervals – this is an important assumption as often the peaks can be quite narrow and can be missed by more general search algorithms. In the second case rounding error makes the computed criterion multimodal even when it is truely unimodal.

In light of these conclusions we see ther are still many loose ends which we would like to tie up. Some of the most important of these are:

The full problem At present the methods for the full problem are only in their infancy, and need to be developed further when a suitable test problem appears. The search for alternative models is ongoing, and as the functions we are considering are very expensive, it might prove cheaper to develop individual models based on the "scientists" actual beliefs.

Varying the covariance structure We need to examine further the differences and similarities between different covariance structures and the subsequent designs –

this question can only really be answered again when we have real problems to work on.

The choice of θ The choice of θ is an important question, we have noted some ways of eliciting prior beliefs about it, and estimating it using $CVMSE$ estimation – but this method does not work satisfactorily for the small number of data points we are considering, and we would like some way of indicating that the prior beliefs are mis-specified.

Diagnostics In the simple case we can often see just by looking at graphs of the predictor that certain of our beliefs have been mis-specified, but when we consider the full problem with multi-dimensional input and output, we cannot easily observe these mis-specifications, and we need some diagnostic tools to help.

Appendix A

Additional Linear Bayes Results and Examples

This appendix completes the notes on the Bayes linear methods of Chapter 3.

A.1 Trajectories

The bearing Y^* produces an overall summary for the updating. This adjustment often comes from various aspects of the data, so it would be useful to break down the bearing into a series of smaller components, to see how these aspects give information. For example how the individual items of information complement or contradict each other.

Let us suppose the data we have contains two separate pieces of information, which we will label these J and K. If we revise our beliefs after observing both J and K, we can obtain the bearing (which we shall label Y_{J+K}). We could also write this revision as occurring in two stages, observing J first and then observing K. After the first stage we have bearing Y_J, and after the second stage we have bearing Y_{J+K}, which is equal to Y_J plus the change in bearing obtained by observing K after observing J,

which we write as $Y_{[K/J]}$. If we look at the lengths of these bearings we have

$$L^2_{J+K} = L^2_J + L^2_{[K/J]} + 2(Y_J, Y_{[K/J]})$$

So in adding the extra evidence K, $L_{[K/J]}$ expresses the size of the extra adjustment in belief having already observed J, and $(Y_J, Y_{[K/J]})$ shows the degree of conflict or support between the two pieces of evidence. If the latter term is large and positive, the two pieces of data are complementary, but if it is large but negative then they are contradictory.

This notion can be extended to subdividing the data into m sections so we obtain a sequence of m belief revisions. (Alternatively we could consider updating our beliefs at m time points, for example this can be compared to the way horse-racing odds change up to the time of the race.) We therefore have a sequence of sets of previsions $\mathrm{P}(\cdot), \mathrm{P}_{[1]}(\cdot), \ldots, \mathrm{P}_{[m]}(\cdot)$, where $\mathrm{P}^*(\cdot) = \mathrm{P}_{[m]}(\cdot)$. Then for each $\mathrm{P}_{[i]}(\cdot)$ we can construct the bearing $Y_{[i]}$, and for each pair i, j we can compute the change in bearing between stage i and stage j.

Defn. A.1 *We call the difference* $Y_{[j/i]} = Y_{[j]} - Y_{[i]}$ *the* bearing for $\mathrm{P}_{[j]}$ adjusted for $\mathrm{P}_{[i]}$.

We then obtain the following results.

Corollary A.1 *For every* X *in* $\mathcal{L}_C$, *and for each* $i < j$,

$$\mathrm{P}_{[j]}(X) - \mathrm{P}_{[i]}(X) = (X, Y_{[j/i]})$$

Corollary A.2 *For every* X *in* $\mathcal{L}_C$, *and for each* $i < j$, $|\mathrm{P}_{[j]}(X) - \mathrm{P}_{[i]}(X)|/\sqrt{\mathrm{Var}(X)}$ *has the maximum value of* $\|Y_{[j/i]}\|$.

Of particular interest to us is the set of one step revisions $Y_{[i/i-1]}$ which we shall write in short hand as $Y_{[i/]}$. Then

Defn. A.2 *A sequence of adjusted bearings* $Y_{[1]}, Y_{[2/]}, \ldots, Y_{[m/]}$ *is a* trajectory over C.

We have noticed that the length (which is the root of its variance) of each bearing is linked to the magnitude of change of belief, and the inner product (or covariance) of two bearings is a measure of the conflict/support between them we can define the following quantities to summarise the trajectory; let $V[i] = \|Y_{[i]}\|^2$, $V[i/] = \|Y_{[i/]}\|^2$, and $C[i] = (Y_{[i-1]}, Y_{[i/]})$, then for each j

$$Y_{[j]} = Y_{[1]} + Y_{[2/]} + \cdot + Y_{[j/]}$$

so

$$V[j] = V[1] + V[2/] + \cdot + V[j/] + 2(C[2] + \cdot + C[j])$$

So to examine how the individual effects of each stage of the revision, we must look at (a) the individual adjusted bearings $Y_{[i/]}$ (those with large lengths, identify stages where there is a large revision) and (b) the way the raw bearing (not adjusted) for the $(i-1)^{\text{st}}$ stage interacts with the adjusted bearing for the i^{th} stage (this is summarised by the magnitude of $C[i]$). A useful summary of the trajectory is given by the pairs $(V[i/], C[i])$. Therefore we make the following definition

Defn. A.3 *Putting* $Cr[i] = \text{Corr}(Y_{[i/]}, Y_{[i-1]}) = \frac{\sqrt{(Y_{[i/]}, Y_{[i-1]})}}{\|Y_{[i/]}\|\|Y_{[i-1]}\|}$, *we define the* route *of the trajectory to be the set of pairs* $(V[i/], Cr[i])$.

Now we can combine the ideas of a trajectory with the notions of belief structure adjustment by projection in Sections 3.5 and 3.6. If, between time t and t^*,we observe a set of data, $\{X_1, \ldots, X_n\}$ and use it as the base for a belief structure D, we can choose $\text{P}^*(\cdot) = \text{P}_d(\cdot)$, the value $\text{P}_D(\cdot)$ takes when we observe $D = d$. We can then compute the data bearing related to this adjustment, which we label Y_d. This belief structure can be partitioned as $\{D[1], \ldots, D[m]\}$, and we can produce a trajectory by adjusting B by $D[1]$, $D[1] + D[2]$, ...

Defn. A.4 *If we adjust our beliefs by projection on a data set, by progressively projecting on larger and larger subsets of the data then the resulting trajectory is termed a* data trajectory.

For our regression example we obtain data trajectories summarised in Table A.1, whose routes are summarised in Table A.2. Looking at these we see some anomalies: the very large value of $V[4/]$ for data set 3 – which highlights the outlier; the large value of $V[5/]$ for data set 4 – which highlights one of the outliers; the large value of $V[1]$, followed by small values for all the $V[i/]$s for data set 6 – shows that the prior specifications of -5 and 5 for a and b are not reflected in the data, but that it is approximately a straight line.

We can compute the expected length of the bearings and adjusted bearings using the cumulative traces of the transforms, as in Section 3.8.

In our example we have the cumulative traces listed in Table A.3, in this we see that in data sets 1–6 the cumulative trace grows at a steady rate as the data is spread evenly across the area, whereas in data sets seven and eight, the individual regions show up, where the transform suddenly jumps, in set 7 after the sixth data point and in set 8 after the tenth data point.

A.2 Raw and pure trajectories

As well as the data trajectory, another trajectory of interest can be constructed by partitioning the data, and then finding the bearing of each set of data individually.

Defn. A.5 *If we take the partition of D, $\{D[1], \ldots, D[t]\}$, we can compute the set of bearings $\{Y_{\{1\}}, \ldots, Y_{\{t\}}\}$ by adjusting our beliefs by $D[i]$. This set of bearings is termed a* raw trajectory.

It would be useful if we could choose the $D[i]$s so that the raw trajectory was also a trajectory (as defined in Section A.1), an automatic way to choose such trajectories is

i	Data set 1	Data set 2	Data set 3
1	$-0.4550-0.0260A+0.0650B$	$0.3063+0.0175A-0.0438B$	$-0.4550-0.0260A+0.0650B$
2	$0.1087+0.1304A+0.1087B$	$0.1071+0.1285A+0.1071B$	$0.1087+0.1304A+0.1087B$
3	$0.1744-0.2267A-0.2616B$	$0.1846-0.2399A-0.2768B$	$0.1744-0.2267A-0.2616B$
4	$-0.3593+0.2723A+0.3442B$	$-0.3425+0.2596A+0.3281B$	$-0.5809+0.4402A+0.5564B$
5	$0.4340-0.2628A-0.3496B$	$0.4575-0.2770A-0.3685B$	$0.5683-0.3441A-0.4577B$
6	$0.1602-0.0832A-0.1152B$	$0.1768-0.0918A-0.1271B$	$0.2420-0.1256A-0.1740B$
7	$-0.1797+0.0819A+0.1178B$	$-0.1693+0.0771A+0.1110B$	$-0.1364+0.0621A+0.0894B$
8	$0.0097-0.0039A-0.0058B$	$0.0160-0.0065A-0.0097B$	$0.0308-0.0124A-0.0186B$
9	$-0.0244+0.0087A+0.0136B$	$-0.0177+0.0063A+0.0099B$	$-0.0151+0.0054A+0.0084B$
10	$0.1698-0.0537A-0.0876B$	$0.1736-0.0549A-0.0896B$	$0.1729-0.0547A-0.0892B$
11	$-0.1629+0.0454A+0.0779B$	$-0.1587+0.0442A+0.0759B$	$-0.1630+0.0454A+0.0780B$

i	Data set 4	Data set 5	Data set 6
1	$0.3063+0.0175A-0.0438B$	$0.0875+0.0050A-0.0125B$	$2.4938+0.1425A-0.3563B$
2	$0.1071+0.1285A+0.1071B$	$0.0542+0.0650A+0.0542B$	$-0.0371-0.0445A-0.0371B$
3	$0.1846-0.2399A-0.2768B$	$0.2447-0.3181A-0.3670B$	$0.0717-0.0931A-0.1075B$
4	$-0.3425+0.2596A+0.3281B$	$-0.3765+0.2854A+0.3607B$	$-0.0330+0.0250A+0.0316B$
5	$0.6010-0.3639A-0.4841B$	$0.4046-0.2450A-0.3259B$	$0.1023-0.0619A-0.0824B$
6	$0.1006-0.0522A-0.0724B$	$0.1143-0.0593A-0.0821B$	$0.0203-0.0106A-0.0146B$
7	$-0.2138+0.0974A+0.1402B$	$-0.2469+0.1125A+0.1619B$	$-0.0573+0.0261A+0.0376B$
8	$-0.0090+0.0036A+0.0054B$	$-0.0695+0.0280A+0.0419B$	$0.1128-0.0455A-0.0680B$
9	$-0.1974+0.0705A+0.1100B$	$-0.1161+0.0415A+0.0647B$	$0.0891-0.0318A-0.0497B$
10	$0.2270-0.0718A-0.1171B$	$0.0714-0.0226A-0.0368B$	$-0.0012+0.0004A+0.0006B$
11	$-0.1223+0.0341A+0.0585B$	$-0.2674+0.0745A+0.1279B$	$-0.1223+0.0341A+0.0585B$

i	Data set 7	Data set 8
1	0	$0.0987+0.0066A-0.0132B$
2	$-0.1000-0.0067A+0.0133B$	$-0.0487-0.0032A+0.0065B$
3	$0.0710-0.0122A-0.0264B$	$0.0002+0.0000A-0.0000B$
4	$0.0251+0.0097A+0.0047B$	$-0.0502-0.0033A+0.0067B$
5	$0.0074-0.0106A-0.0120B$	$-0.0605-0.0040A+0.0080B$
6	$0.0155-0.0223A-0.0254B$	$-0.1352+0.0418A+0.0688B$
7	$0.0177-0.0095A-0.0130B$	$0.0022-0.0007A-0.0011B$
8	$0.0173-0.0090A-0.0125B$	$0.0292-0.0261A-0.0319B$
9	$0.0381-0.0199A-0.0275B$	$-0.0356+0.0317A+0.0388B$
10	$0.0020-0.0010A-0.0014B$	$-0.0020+0.0018A+0.0022B$
11	$0.0219-0.0107A-0.0150B$	$0.0947-0.0513A-0.0702B$

Table A.1: Trajectories for data sets

i	Data set 1			Data set 2			Data set 3		
	$V[i]$	$V[i/]$	$Cr[i]$	$V[i]$	$V[i/]$	$Cr[i]$	$V[i]$	$V[i/]$	$Cr[i]$
1	0.1325			0.0600			0.1325		
2	0.4398	0.4610	-0.3110	0.6096	0.4477	0.3110	0.4398	0.4610	-0.3110
3	0.4043	1.5221	-0.9519	0.5657	1.7045	-0.8576	0.4043	1.5221	-0.9519
4	0.9381	2.3158	-0.9208	0.7426	2.1044	-0.8833	3.3755	6.0518	-0.9208
5	0.3446	2.2364	-0.9769	0.7993	2.4851	-0.8938	0.0332	3.8344	-0.9961
6	1.0972	0.2314	0.9229	1.9507	0.2818	0.9162	0.6613	0.5281	0.3774
7	0.3518	0.2318	-0.9689	0.9218	0.2057	-0.9745	0.2178	0.1336	-0.9709
8	0.3768	0.0005	0.8841	0.9944	0.0015	0.9556	0.2854	0.0055	0.8942
9	0.3223	0.0028	-0.8754	0.9211	0.0015	-0.9685	0.2545	0.0011	-0.9057
10	0.7544	0.1138	0.8313	1.6855	0.1189	0.9754	0.6756	0.1179	0.8751
11	0.3737	0.0864	-0.9148	1.0293	0.0820	-0.9928	0.3071	0.0865	-0.9410

i	Data set 4			Data set 5			Data set 6		
	$V[i]$	$V[i/]$	$Cr[i]$	$V[i]$	$V[i/]$	$Cr[i]$	$V[i]$	$V[i/]$	$Cr[i]$
1	0.0600			0.0049			3.9800		
2	0.6096	0.4477	0.3110	0.1342	0.1145	0.3110	3.7462	0.0537	-0.3110
3	0.5657	1.7045	-0.8576	1.9718	2.9953	-0.9131	4.0722	0.2569	0.0352
4	0.7426	2.1044	-0.8833	0.0385	2.5429	-0.9995	3.8909	0.0195	-0.3562
5	1.8408	4.2882	-0.8938	1.4550	1.9436	-0.9640	4.4804	0.1241	0.3348
6	2.7255	0.0913	-0.9679	2.4005	0.1178	1.0000	4.6157	0.0037	0.5087
7	1.1909	0.3283	-0.9847	0.7888	0.4377	-1.0000	4.2715	0.0236	-0.5571
8	1.1452	0.0005	-0.9729	0.5196	0.0282	-0.9969	4.9465	0.0742	0.5336
9	0.4261	0.1865	-0.9797	0.2211	0.0646	-0.9911	5.5450	0.0380	0.6462
10	1.1925	0.2033	0.9566	0.3075	0.0201	0.9701	5.5367	0.0000	-0.7210
11	0.7633	0.0487	-0.9914	0.0317	0.2329	-0.9731	5.6515	0.0010	0.7502

i	Data set 7			Data set 8		
	$V[i]$	$V[i/]$	$Cr[i]$	$V[i]$	$V[i/]$	$Cr[i]$
1	0.0000			0.0064		
2	0.0066	0.0066	0.0000	0.0016	0.0016	-1.0000
3	0.0096	0.0088	-0.3778	0.0017	0.0000	1.0000
4	0.0023	0.0027	-0.9877	0.0000	0.0017	-1.0000
5	0.0109	0.0033	0.9863	0.0024	0.0024	0.0000
6	0.0508	0.0146	0.9972	0.0762	0.0696	0.1623
7	0.0779	0.0030	0.9819	0.0739	0.0000	-0.9845
8	0.1095	0.0027	0.9873	0.0235	0.0208	-0.9076
9	0.1982	0.0133	0.9909	0.0903	0.0000	0.6684
10	0.2034	0.0000	0.9932	0.0960	0.0001	0.9251
11	0.2630	0.0038	0.9934	0.0075	0.0872	-0.9603

Table A.2: Routes for data sets

i	Data sets 1–6		Data sets 7		Data set 8	
	$P_{\mathbf{Y}}(V[i])$	$P_{\mathbf{Y}}(V[i/])$	$P_{\mathbf{Y}}(V[i])$	$P_{\mathbf{Y}}(V[i/])$	$P_{\mathbf{Y}}(V[i])$	$P_{\mathbf{Y}}(V[i/])$
1	0.9800		0.9737		0.9737	
2	1.0613	0.0813	0.9867	0.0130	0.9867	0.0130
3	1.2557	0.1943	1.0196	0.0329	0.9911	0.0044
4	1.5000	0.2443	1.0254	0.0058	0.9933	0.0022
5	1.6960	0.1960	1.0781	0.0526	0.9946	0.0013
6	1.8194	0.1234	1.1121	0.0341	1.0314	0.0367
7	1.8902	0.0708	1.9076	0.7955	1.0550	0.0237
8	1.9302	0.0400	1.9519	0.0442	1.1105	0.0555
9	1.9532	0.0229	1.9663	0.0145	1.1510	0.0404
10	1.9667	0.0135	1.9741	0.0078	1.1818	0.0308
11	1.9748	0.0081	1.9786	0.0044	1.9196	0.7379

Table A.3: Traces of transforms

Defn. A.6 *For any two belief structures, B and D, we can construct the twin maps $M(B)$ and $M(D)$. Any raw trajectory based on a partition $\{M[1],\ldots,M[t]\}$ of $M(D)$ is called a* pure trajectory $\{Y_{\{1\}},\ldots,Y_{\{t\}}\}$.

We summarise properties of the pure trajectory in the following theorem

Theorem A.3 *A pure trajectory satisfies*

(a) For any observed set of outcomes the data and raw trajectories are equivalent.

(b) The bearings are uncorrelated, and the squared length of the bearing corresponding to the subset $M[i_1]+\cdots+M[i_k]$ of $M(D)$ is equal to the sum of the squared lengths of the bearings of $M[i_1],\ldots,M[i_k]$.

(c) The expected squared length of $Y_{\{i\}}$ is equal to the sum of the eigenvalues of T_D whose eigenvectors are in $M[i]$.

This can then be used to construct useful trajectories, by first splitting the eigenstructure of the transform into useful subsets, and examining the bearing of these subsets.

A.3 Generalised belief transforms

Finally before I go onto some examples it should be pointed out that the idea of belief transform in the previous sections does not need to be restricted to adjustment of previsions by data. If we have a belief structures B, with two inner products $(\cdot,\cdot)$ and $(\cdot,\cdot)^*$ we can obtain a bounded positive symmetric functional S such that $(X,Y)^* = (X,S(Y))$, and we call S a *generalised belief transform.* Compare this with Property 3.15 of belief transforms from Section 3.8. It does not have all the properties of the belief transform, its eigenvalues are no longer bounded above by one.

As well as S we can define the inverse transform S^{-1}, by $(X,Y) = (X,S^{-1}(Y))^*$ providing we restrict it to the strictly positive part of $(\cdot,\cdot)^*$. The eigenvectors of S^{-1} (with strictly positive eigenvalues) are the same as those of S, and the corresponding eigenvalue of S^{-1} is the reciprocal of the eigenvalue of S.

This notion can then be used to compare belief structures for such topics as hypothesis testing, experimental design and sensitivity analysis where we want to compare different stochastic models of the quantities.

A.4 Some more simple examples

A.4.1 Example 1

We would like to refine our estimate of a quantity X, by making repeated observations $\{X_1,\ldots,X_n\}$. *A priori*, we express our prevision of X as μ and our prevision of X^2 as $\mu^2+\sigma^2$. We assume that each observation can be represented by $X_i = X + Z_i$, where the Z_is are uncorrelated with zero mean and variance ε^2, and are independent of X. Hence

$$\begin{aligned} \mathrm{P}(X_i) &= \mathrm{P}(X) + \mathrm{P}(Z_i) \\ &= \mu, \text{ for } 1 \leq i \leq n \\ \mathrm{P}(X_i X) &= \mathrm{P}(X^2) + \mathrm{P}(XZ_i) \end{aligned}$$

$$\begin{aligned} &= \mu^2 + \sigma^2, \text{ for } 1 \leq i \leq n \\ \mathrm{P}(X_i^2) &= \mathrm{P}(X^2) + 2\mathrm{P}(XZ_i) + \mathrm{P}(Z_i^2) \\ &= \mu^2 + \sigma^2 + \varepsilon^2, \text{ for } 1 \leq i \leq n \\ \mathrm{P}(X_iX_j) &= \mathrm{P}(X^2) + \mathrm{P}(XZ_i) + \mathrm{P}(XZ_j) + \mathrm{P}(Z_iZ_j) \\ &= \mu^2 + \sigma^2, \text{ for } 1 \leq i \neq j \leq n \end{aligned}$$

We therefore construct two belief structures, B with base $\{1, X\}$, and D with base $\{1, X_1, \ldots, X_n\}$. $P_D(X)$ can be computed – using equation (3.4) – as

$$\mathrm{P}_D(X) = \mu + \sigma^2\mathbf{1}^{\mathrm{T}}(\sigma^2\mathbf{1}\mathbf{1}^{\mathrm{T}} + \varepsilon^2 I)^{-1}(\boldsymbol{X} - \mu\mathbf{1}) \tag{A.1}$$

and the adjusted variance

$$\mathrm{Var}([X/D]) = \sigma^2 - \sigma^2\mathbf{1}^{\mathrm{T}}(\sigma^2\mathbf{1}\mathbf{1}^{\mathrm{T}} + \varepsilon^2 I)^{-1}\sigma^2\mathbf{1} \tag{A.2}$$

and then inverting the variance matrix using the identity $(Q + LL^{\mathrm{T}})^{-1}L = Q^{-1}L(I + L^{\mathrm{T}}Q^{-1}L)^{-1}$ to obtain

$$\begin{aligned} \mathrm{P}_D(X) &= \frac{n\sigma^2\overline{X} + \varepsilon^2\mu}{\varepsilon^2 + n\sigma^2} \\ \mathrm{Var}([X/D]) &= \frac{\sigma^2\varepsilon^2}{\varepsilon^2 + n\sigma^2} \end{aligned}$$

where $\overline{X}$ is the sample mean $n^{-1}\sum_{i=1}^{n} X_i$. We note that the prevision is a weighted average of the sample and prior means. The matrices representing P_B and P_D are

$$\begin{aligned} P_D &= \begin{pmatrix} 1 & \frac{\varepsilon^2\mu}{\varepsilon^2+n\sigma^2} \\ \mathbf{0} & \frac{\sigma^2}{\varepsilon^2+n\sigma^2}\mathbf{1} \end{pmatrix} \\ P_B &= \begin{pmatrix} 1 & \mathbf{0}^{\mathrm{T}} \\ 0 & \mathbf{1}^{\mathrm{T}} \end{pmatrix} \end{aligned}$$

and the corresponding belief transforms are

$$T_B = \begin{pmatrix} 1 & \frac{\varepsilon^2\mu}{\varepsilon^2+n\sigma^2}\mathbf{1} \\ \mathbf{0} & \frac{\sigma^2}{\varepsilon^2+n\sigma^2}\mathbf{1}\mathbf{1}^{\mathrm{T}} \end{pmatrix}$$

$$T_D = \begin{pmatrix} 1 & \frac{\varepsilon^2\mu}{\varepsilon^2+n\sigma^2} \\ 0 & \frac{n\sigma^2}{\varepsilon^2+n\sigma^2} \end{pmatrix}$$

The eigenanalysis of T_D is straight forward, the eigenvalues being $\lambda_0 = 1$ and $\lambda_1 = \frac{n\sigma^2}{\varepsilon^2+n\sigma^2}$, with eigenvectors $E_0 = 1$ and $E_1 = \frac{X-\mu}{\sigma}$. Then by using Lemma 3.4 T_B can be shown to have eigenvalues $\lambda_0 = 1, \lambda_1 = \frac{n\sigma^2}{\varepsilon^2+n\sigma^2}, \lambda_2 = \cdots = \lambda_n = 0$ with $F_0 = 1$ and $F_1 = \frac{\overline{X}-\mu}{\sqrt{\varepsilon^2/n+\sigma_2}}$. The eigenspace corresponding to the zero eigenvalues is spanned by the residuals $X_i - \overline{X}$. We can write down the bearing of the data,

$$\begin{aligned} Y_d &= \frac{\sigma}{\sqrt{\varepsilon^2/n+\sigma^2}} \frac{\overline{X}-\mu}{\sqrt{\varepsilon^2/n+\sigma_2}} \frac{X-\mu}{\sigma} \\ &= \frac{\overline{X}-\mu}{\varepsilon^2/n+\sigma^2}(X-\mu) \end{aligned}$$

To construct the data trajectory we compute the bearings

$$\begin{aligned} Y_{[i]} &= \frac{\overline{X}_i-\mu}{\varepsilon^2/i+\sigma^2}(X-\mu) \\ Y_{[i/]} &= \frac{(X_i-\mu)\varepsilon^2+(i-1)(X_i-\overline{X}_{i-1})\sigma^2}{(\varepsilon^2+i\sigma^2)(\varepsilon^2+(i-1)\sigma^2)}(X-\mu) \end{aligned}$$

where $\overline{X}_i = i^{-1}\sum_{j=1}^{i} X_j$ is the cumulative sample mean. The variance $V[i]$ and $V[i/]$ and correlations $Cr[i]$ can be computed,

$$\begin{aligned} V[i] &= \left\{\frac{\sigma(\overline{X}_i-\mu)}{\varepsilon^2/i+\sigma^2}\right\}^2 \\ V[i/] &= \left\{\frac{(X_i-\mu)\varepsilon^2+(i-1)(X_i-\overline{X}_{i-1})\sigma^2}{(\varepsilon^2+i\sigma^2)(\varepsilon^2+(i-1)\sigma^2)}\right\}^2 \sigma^2 \\ Cr[i] &= \begin{cases} +1 & \text{where } \overline{X}_i-\mu \text{ and } \mathrm{P}_{X_1,\ldots,X_i}(X) - \mathrm{P}_{X_1,\ldots,X_{i-1}}(X) \text{ have the same sign} \\ -1 & \text{where } \overline{X}_i-\mu \text{ and } \mathrm{P}_{X_1,\ldots,X_i}(X) - \mathrm{P}_{X_1,\ldots,X_{i-1}}(X) \text{ have different signs} \end{cases} \end{aligned}$$

to produce the data-trajectory. We therefore see that all the data perfectly complements or completely contradicts the previous data. We can compare $V[i]$ with the trace of the transforms $\text{trace}(T_{D[1]+\ldots+D[i]}) - 1 = \frac{i\sigma^2}{\varepsilon^2+i\sigma^2}$, and $V[i/]$ with the difference between successive traces $\text{trace}(T_{D[1]+\ldots+D[i]}) - \text{trace}(T_{D[1]+\ldots+D[i-1]}) = \frac{\sigma^2\varepsilon^2}{(\varepsilon^2+i\sigma^2)(\varepsilon^2+(i-1)\sigma^2)}$ to look for any irregularities in the data.

If for example we elicit that $\mu = 0$, $\varepsilon^2 = 25$ and $\sigma = 1$, the prevision of the squared lengths are $i/(i+25)$, and the prevision of the adjusted lengths are $25/(25+i)(24+i)$. So if we observe the following data set, $d = \{3, 7, -3, 8, -4, -20, 3, -7, 2, -2\}$, we obtain the trajectory summarised in table A.4. The possibly anomalous result of $d_6 = -20$ is highlighted by a large value of $V[6/]$ which is of an order of magnitude greater than any of the other $V[i/]$ values, and sixteen times greater than its expectation.

i	d_i	$V[i]$	$V[i/]$	$Cr[i]$	P($V[i]$)	P($V[i/]$)
1	3	0.013	–	–	0.038	–
2	7	0.137	0.065	1.0	0.074	0.046
3	-3	0.063	0.014	-1.0	0.107	0.033
4	8	0.2684	0.071	1.0	0.138	0.031
5	-4	0.134	0.023	-1.0	0.167	0.029
6	-20	0.084	0.432	-1.0	0.194	0.027
7	3	0.035	0.011	-1.0	0.219	0.025
8	-7	0.155	0.043	1.0	0.242	0.024
9	2	0.105	0.005	-1.0	0.265	0.022
10	2	0.138	0.002	1.0	0.286	0.021

Table A.4: Trajectory for example 1

A.4.2 Example 2 – Poincaré's problem.

We have an unknown function $Y(x)$ which we wish to approximate in some way. To do this we assume that $Y(x)$ can be written as a convergent power series,

$$Y(x) = A_0 + A_1 x + A_2 x^2 + \cdots.$$

We then specify our beliefs about the As; for simplicity we believe that they are uncorrelated, and have zero prevision, the latter assumption is made because we have no knowledge of the sign of the A_i. We summarize of our previsions:

$$\begin{aligned}
\mathrm{P}(A_i) &= 0, \text{for } 0 \leq i \\
\mathrm{P}(A_i A_j) &= \sigma^2 g_i \delta_{ij}, \text{for } 0 \leq i, j \\
\mathrm{P}(Y(x)) &= \mathrm{P}(A_0) + \mathrm{P}(A_1)x + \mathrm{P}(A_2)x^2 + \cdots = 0 \\
\mathrm{P}(A_i Y(x)) &= \sum_{j=0}^{\infty} \mathrm{P}(A_i A_j) x^j = \sigma^2 g_i x^i, \text{for } 0 \leq i \\
\mathrm{P}(Y(x)Y(w)) &= \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} \mathrm{P}(A_i A_j) x^i w^j = \sum_{i=0}^{\infty} \sigma^2 g_i (xw)^i
\end{aligned}$$

where the g_is are chosen to express our beliefs. Then if we put $g(t) = \sum_{i=0}^{\infty} g_i t^i$, we can write $\mathrm{P}(Y(x)Y(w)) = g(xw)$.

We them obtain our estimates for A_i, and $Y(x_*)$, if we observe $y_1, \ldots, y_n$ at $x_1, \ldots, x_n$

$$\begin{aligned}
\mathrm{P}_D(A_i) &= g_i \boldsymbol{x}_i G^{-1} \boldsymbol{y} \\
\mathrm{P}_D(Y(x^*)) &= \boldsymbol{g} G^{-1} \boldsymbol{y}
\end{aligned}$$

where $\boldsymbol{x}_i = (x_1^i, \ldots, x_n^i)^{\mathrm{T}}$, $G_{ij} = g(x_i x_j)$ and $\boldsymbol{g} = (g(x_1 x^*), \ldots, g(x_n x^*))^{\mathrm{T}}$. We can now construct two belief structures, B consisting of the unit constant 1 and the coefficients of the approximation A_i and D consisting of 1 and $Y_j = Y(x_j)$ observations of the unknown function at a series of design points $\{x_1, \ldots, x_n\}$. As there is no error, knowing all the coefficients means we know the exact function, so T_B the belief transform obtained by adjusting D by B is the identity transform, so has $n+1$ eigenvalues all equal to 1, with eigenvectors $F_0 = 1$, $F_i = (\sigma^2 g(x_i^2))^{-1/2} Y_i$. Then using Lemma 3.4 the non-zero eigenvalues of T_D are $n+1$ ones, with eigenvectors $E_0 = 1$, $E_i = (\sigma^2 g(x_i^2))^{-1/2} \sum_{j=0}^{\infty} A_j x_i^j$.

A.4.3 Example 3 – exchangeable belief structures.

We have a collection of beliefs about quantities $\{X_1, \ldots, X_r\}$ which we wish to modify by making some observations. We have s observables $\{Y_{(1)}, \ldots, Y_{(s)}\}$, which we can make repeated These observations are to be made in groups $\{Y_{i1}, \ldots, Y_{is}\}$ for i in $1, \ldots, n$. These sets of observations are exchangeable, i.e.

$$\begin{aligned} \mathrm{P}(Y_{ij}) &= m_j \quad \forall \quad i,j \\ \mathrm{P}(Y_{ij}Y_{iJ}) &= m_j m_J + v_{jJ} \quad \forall \quad i,j,J \\ \mathrm{P}(Y_{ij}Y_{IJ}) &= m_j m_J + u_{jJ} \quad \forall \quad i \neq I, j, J \end{aligned}$$

and they are coexchangeable with $\{X_1, \ldots, X_r\}$, i.e.

$$\mathrm{P}(X_k Y_{ij}) = \mu_k m_j + c_{kj} \quad \forall \quad i,j,k$$

whose second order structure is

$$\begin{aligned} \mathrm{P}(X_k) &= \mu_k \quad \forall \quad k \\ \mathrm{P}(X_k X_K) &= \mu_k \mu_K + \sigma_{kK} \quad \forall \quad k, K \end{aligned}$$

We now define the two belief structures B and D with bases $\{X_1, \ldots, X_r\}$ and $\{Y_{11}, \ldots, Y_{1s}, \ldots, Y_{n1}, \ldots, Y_{ns}\}$ respectively. We can then compute P_B and P_D (Notation $W = V - U$)

$$P_B = \begin{bmatrix} 1 & \mathbf{1}^{\mathrm{T}} \otimes \left(\boldsymbol{m}^{\mathrm{T}} - \boldsymbol{\mu}^{\mathrm{T}} \Sigma^{-1} C\right) \\ \mathbf{0} & \mathbf{1}^{\mathrm{T}} \otimes (\Sigma^{-1} C) \end{bmatrix}$$

$$P_D = \begin{bmatrix} 1 & \boldsymbol{\mu}^{\mathrm{T}} - \left(\mathbf{1}^{\mathrm{T}} \otimes \boldsymbol{m}^{\mathrm{T}}\right) \left(\mathbf{1}\mathbf{1}^{\mathrm{T}} \otimes U + I \otimes W\right)^{-1} \left(\mathbf{1} \otimes C^{\mathrm{T}}\right) \\ \mathbf{0} & \left(\mathbf{1}\mathbf{1}^{\mathrm{T}} \otimes U + I \otimes W\right)^{-1} \left(\mathbf{1} \otimes C^{\mathrm{T}}\right) \end{bmatrix}$$

$$= \begin{bmatrix} 1 & \boldsymbol{\mu}^{\mathrm{T}} - n\boldsymbol{m}^{\mathrm{T}}\left(\left[W^{-1} - nW^{-1}U\left(nU+W\right)^{-1}\right]C^{\mathrm{T}}\right) \\ \mathbf{0} & \mathbf{1}\otimes\left(\left[W^{-1} - nW^{-1}U\left(nU+W\right)^{-1}\right]C^{\mathrm{T}}\right) \end{bmatrix}$$

$$= \begin{bmatrix} 1 & \boldsymbol{\mu}^{\mathrm{T}} - n\boldsymbol{m}^{\mathrm{T}}\left([n-1]U+V\right)^{-1}C^{\mathrm{T}} \\ \mathbf{0} & \mathbf{1}\otimes\left[\left([n-1]U+V\right)^{-1}C^{\mathrm{T}}\right] \end{bmatrix}$$

$$T_B = \begin{bmatrix} 1 & \mathbf{1}^{\mathrm{T}}\otimes\left(\boldsymbol{m}\left[I - n\left([n-1]U+V\right)^{-1}C^{\mathrm{T}}\Sigma^{-1}C\right]\right) \\ \mathbf{0} & \left[\mathbf{1}\mathbf{1}^{\mathrm{T}}\right]\otimes\left[\left([n-1]U+V\right)^{-1}C^{\mathrm{T}}\Sigma^{-1}C\right] \end{bmatrix}$$

$$T_D = \begin{bmatrix} 1 & \boldsymbol{\mu}^{\mathrm{T}}\left[I - n\Sigma^{-1}C\left([n-1]U+V\right)^{-1}C^{\mathrm{T}}\right] \\ \mathbf{0} & n\Sigma^{-1}C\left([n-1]U+V\right)^{-1}C^{\mathrm{T}} \end{bmatrix}$$

Using these results we make the following observations:

Observation A.1 *The heart of the transform (the sufficient statistics) is made up of linear combinations of the averages of the observations (averaging over the is), i.e.* $Y_{\cdot j} = n^{-1}\sum_{i=1}^{n} Y_{ij}$

Proof

The eigenvectors of T_B are given in Lemma 3.4 by $\mathrm{P}_B(E_i)$ where the E_is are the eigenvectors of T_D, and so have the form $\mathbf{1}\otimes\boldsymbol{f} = \sum_{j=1}^{s} f_j \sum_{i=1}^{n} Y_{ij}$.

□

Observation A.2 *The number of non-zero eigenvalues is less than the minimum of* $r+1$ *and* $s+1$.

Observation A.3 *If* $\boldsymbol{e} = (e_1,\ldots,e_r)^{\mathrm{T}}$ *is an eigenvector of* $\Sigma^{-1}C\left([n-1]U+V\right)^{-1}C^{\mathrm{T}}$ *corresponding to the eigenvalue* λ, *then*

$$E = (-e_1\mu_1 - \cdots - e_r\mu_r, e_1, \ldots, e_r)^{\mathrm{T}} = e_1(X_1-\mu_1) + \cdots + e_r(X_r-\mu_r)$$

is an eigenvector of T_D *corresponding to the eigenvalue* $n\lambda$.

Observation A.4 *If* $\boldsymbol{f} = (f_1, \ldots, f_s)^{\mathrm{T}}$ *is an eigenvector of* $([n-1]U + V)^{-1} C^{\mathrm{T}} \Sigma^{-1} C$ *corresponding to the eigenvalue* λ*, then*

$$\begin{aligned} F &= (-n(f_1 m_1 + \cdots + f_s m_s), f_1, \ldots, f_s, \ldots, f_1, \ldots, f_s)^{\mathrm{T}} \\ &= f_1(\sum_{i=1}^{n} Y_{i1} - n m_1) + \cdots + f_s(\sum_{i=1}^{n} Y_{is} - n m_s) \end{aligned}$$

is an eigenvector of T_B *corresponding to the eigenvalue* $n\lambda$*.*

Observation A.5 *For all* $\boldsymbol{g} = (g_1, \ldots, g_s)$ *in* $\mathbb{R}^s$*, and for all* $\boldsymbol{a} = (\alpha_1, \ldots, \alpha_n)$ *in* $\mathbb{R}^n$ *st* $\sum_{i=1}^{n} \alpha_i = 0$

$$\begin{bmatrix} 0 \\ \boldsymbol{a} \otimes \boldsymbol{g} \end{bmatrix}$$

is an eigenvectors of T_B *with eigenvalue* 0*.*

Moreover, if $G = \begin{pmatrix} 0 \\ \boldsymbol{a} \otimes \boldsymbol{g} \end{pmatrix}$ *and* $H = \begin{pmatrix} 0 \\ \boldsymbol{b} \otimes \boldsymbol{h} \end{pmatrix}$ *then*

$$\begin{aligned} (G, H) &= (\boldsymbol{a} \otimes \boldsymbol{g})^{\mathrm{T}} \left((\mathbf{1}\mathbf{1}^{\mathrm{T}}) \otimes (\boldsymbol{m}\boldsymbol{m}^{\mathrm{T}} + U) + I \otimes [V - U] \right) (\boldsymbol{b} \otimes \boldsymbol{h}) \\ &= \boldsymbol{a}^{\mathrm{T}} \mathbf{1}\mathbf{1}^{\mathrm{T}} \boldsymbol{b} \boldsymbol{g}^{\mathrm{T}} (\boldsymbol{m}\boldsymbol{m}^{\mathrm{T}} + U) \boldsymbol{h} + \boldsymbol{a}^{\mathrm{T}} \boldsymbol{b} \boldsymbol{g}^{\mathrm{T}} [V - U] \boldsymbol{h} \\ &= (\boldsymbol{a}^{\mathrm{T}} \boldsymbol{b})(\boldsymbol{g}^{\mathrm{T}} [V - U] \boldsymbol{h}) \end{aligned}$$

These observations mean we can work out the eigenstructure of the transforms more easily. (Observations A.3 and A.4 show how to find the meaningful eigenvectors, and A.5 shows how to find the residuals). If we assume that the number of replicates n is large, then we can see that

$$\begin{aligned} P_D &\simeq \begin{bmatrix} 1 & \boldsymbol{\mu}^{\mathrm{T}} - \boldsymbol{m} U^{-1} C^{\mathrm{T}} \\ \mathbf{0} & \mathbf{1} \otimes U^{-1} C^{\mathrm{T}} \end{bmatrix} \\ T_D &\simeq \begin{bmatrix} 1 & \boldsymbol{\mu}^{\mathrm{T}} (I - \Sigma^{-1} C U^{-1} C^{\mathrm{T}}) \\ \mathbf{0} & \Sigma^{-1} C U^{-1} C^{\mathrm{T}} \end{bmatrix} \end{aligned}$$

You will note that the intra-group covariance disappears and the result just depends on the inter-group covariance. This asymptotic result is useful as it means that we do not have to work out the transforms each time we receive a new piece of data.

This model is useful if we are looking at properties of an object which we can take repeated observations on some object or class of objects, for example a car, the X_ks being average speed, maximum speed, petrol consumption etc, and the Y_{ij}s the distances travelled in various times,..., for various cars of the same type.

Appendix B

Omitted Algebra From Chapter 5

B.1 The univariate model – Section 5.2

$$
\begin{aligned}
\mathrm{P}_{\boldsymbol{Y}}(X_0) &= \mu_0 - b\sigma_0^2 \mathbf{1}^{\mathrm{T}}(b^2\sigma_0^2 \mathbf{1}\mathbf{1}^{\mathrm{T}} + \sigma_b^2 C)^{-1}(\boldsymbol{Y} - b\boldsymbol{x} + b\mu_0 \mathbf{1}) \\
&= \mu_0 - \frac{b\sigma_0^2 \mathbf{1}^{\mathrm{T}} C^{-1}(\boldsymbol{Y} - b\boldsymbol{x} + b\mu_0 \mathbf{1})}{\sigma_b^2(1 + \frac{b^2\sigma_0^2}{\sigma_b^2}\mathbf{1}^{\mathrm{T}} C^{-1}\mathbf{1})} \\
&= \frac{\sigma_b^2\mu_0 - \sigma_0^2 \mathbf{1}^{\mathrm{T}} C^{-1}(\boldsymbol{Y} - b\boldsymbol{x})b}{\sigma_b^2 + \sigma_0^2 \mathbf{1}^{\mathrm{T}} C^{-1}\mathbf{1}b^2} \qquad \text{(B.1)}
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{Var}[X_0/\boldsymbol{Y}] &= \sigma_0^2 - b\sigma_0^2 \mathbf{1}^{\mathrm{T}}(b^2\sigma_0^2 \mathbf{1}\mathbf{1}^{\mathrm{T}} + \sigma_b^2 C)^{-1} b\sigma_0^2 \mathbf{1} \\
&= \sigma_0^2 - \frac{b^2\sigma_0^4 \mathbf{1}^{\mathrm{T}} C^{-1}\mathbf{1}}{\sigma_b^2(1 + \frac{b^2\sigma_0^2}{\sigma_b^2}\mathbf{1}^{\mathrm{T}} C^{-1}\mathbf{1})} \\
&= \frac{\sigma_0^2\sigma_b^2}{\sigma_b^2 + \sigma_0^2 \mathbf{1}^{\mathrm{T}} C^{-1}\mathbf{1}b^2} \qquad \text{(B.2)}
\end{aligned}
$$

$$
\begin{aligned}
P_{\boldsymbol{Y}}(B(x_*)) &= b + \sigma_b^2 \boldsymbol{d}^{\mathrm{T}}(b^2\sigma_0^2 \mathbf{1}\mathbf{1}^{\mathrm{T}} + \sigma_b^2 C)^{-1}(\boldsymbol{Y} - b\boldsymbol{x} + b\mu_0 \mathbf{1}) \\
&= b + \frac{\sigma_b^2 \boldsymbol{d}^{\mathrm{T}} C^{-1}(\boldsymbol{Y} - b\boldsymbol{x} + b\mu_0 \mathbf{1})}{\sigma_b^2} - \\
&\qquad \frac{\sigma_b^2 \boldsymbol{d}^{\mathrm{T}} C^{-1}\mathbf{1}b^2\sigma_0^2 \mathbf{1}^{\mathrm{T}} C^{-1}(\boldsymbol{Y} - b\boldsymbol{x} + b\mu_0 \mathbf{1})}{\sigma_b^4(1 + \frac{b^2\sigma_0^2}{\sigma_b^2}\mathbf{1}^{\mathrm{T}} C^{-1}\mathbf{1})} \\
&= b + \boldsymbol{d}^{\mathrm{T}} C^{-1}(\boldsymbol{Y} - b\boldsymbol{x}) +
\end{aligned}
$$

$$
\begin{aligned}
& \frac{\boldsymbol{d}^{\mathrm{T}} C^{-1}\mathbf{1}(b\mu(\sigma_b^2+\sigma_0^2\mathbf{1}^{\mathrm{T}}C^{-1}\mathbf{1}b^2)-b^2\sigma_0^2\mathbf{1}^{\mathrm{T}}C^{-1}(\boldsymbol{Y}-b\boldsymbol{x}+b\mu_0\mathbf{1})}{\sigma_b^2+\sigma_0^2\mathbf{1}^{\mathrm{T}}C^{-1}\mathbf{1}b^2} \\
= \; & b+\boldsymbol{d}^{\mathrm{T}}C^{-1}(\boldsymbol{Y}-b\boldsymbol{x})+\frac{\boldsymbol{d}^{\mathrm{T}}C^{-1}\mathbf{1}b(\sigma_b^2\mu-\sigma_0^2\mathbf{1}^{\mathrm{T}}C^{-1}(\boldsymbol{Y}-b\boldsymbol{x})b)}{\sigma_b^2+\sigma_0^2\mathbf{1}^{\mathrm{T}}C^{-1}\mathbf{1}b^2} \\
= \; & b+\boldsymbol{d}^{\mathrm{T}}C^{-1}(\boldsymbol{Y}-b[\boldsymbol{x}-P_{\boldsymbol{Y}}(X_0)\mathbf{1}]) \qquad \text{(B.3)}
\end{aligned}
$$

$\mathrm{Cov}([B(x_*)/\boldsymbol{Y}],[B(\tilde{x}_*)/\boldsymbol{Y}])$

$$
\begin{aligned}
= \; & \sigma_b^2\rho(x_*-\tilde{x}_*)-\sigma_b^2\boldsymbol{d}^{\mathrm{T}}(b^2\sigma_0^2\mathbf{1}\mathbf{1}^{\mathrm{T}}+\sigma_b^2C)^{-1}\sigma_b^2\tilde{\boldsymbol{d}} \\
= \; & \sigma_b^2\rho(x_*-\tilde{x}_*)-\sigma_b^4\left(\frac{\boldsymbol{d}^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{d}}}{\sigma_b^2}-\frac{\boldsymbol{d}^{\mathrm{T}}C^{-1}\mathbf{1}b^2\sigma_0^2\mathbf{1}^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{d}}}{\sigma_b^4(1+\frac{b^2\sigma_0^2}{\sigma_b^2}\mathbf{1}^{\mathrm{T}}C^{-1}\mathbf{1})}\right) \\
= \; & \sigma_b^2\left(\rho(x_*-\tilde{x}_*)-\boldsymbol{d}^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{d}}+\frac{\sigma_0^2\boldsymbol{d}^{\mathrm{T}}C^{-1}\mathbf{1}\mathbf{1}^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{d}}b^2}{\sigma_b^2+\sigma_0^2\mathbf{1}^{\mathrm{T}}C^{-1}\mathbf{1}b^2}\right) \qquad \text{(B.4)}
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{P}_{\boldsymbol{Y}}(Y(x_*)) = \; & b(x_*-\mu_0)+(b^2\sigma_0^2\mathbf{1}+\sigma_b^2\boldsymbol{c})^{\mathrm{T}}(b^2\sigma_0^2\mathbf{1}\mathbf{1}^{\mathrm{T}}+\sigma_b^2C)^{-1}(\boldsymbol{Y}-b\boldsymbol{x}+b\mu_0\mathbf{1}) \\
= \; & b[x_*-\mathrm{P}_{\boldsymbol{Y}}(X_0)]+\frac{\sigma_b^2\boldsymbol{c}^{\mathrm{T}}C^{-1}(\boldsymbol{Y}-b\boldsymbol{x}+b\mu_0\mathbf{1})}{\sigma_b^2}- \\
& \frac{\sigma_b^2\boldsymbol{c}^{\mathrm{T}}C^{-1}\mathbf{1}b^2\sigma_0^2\mathbf{1}^{\mathrm{T}}C^{-1}(\boldsymbol{Y}-b\boldsymbol{x}+b\mu_0\mathbf{1})}{\sigma_b^4(1+\frac{b^2\sigma_0^2}{\sigma_b^2}\mathbf{1}^{\mathrm{T}}C^{-1}\mathbf{1})} \\
= \; & b[x_*-\mathrm{P}_{\boldsymbol{Y}}(X_0)]+\boldsymbol{c}^{\mathrm{T}}C^{-1}(\boldsymbol{Y}-b\boldsymbol{x})+ \\
& \frac{\boldsymbol{c}^{\mathrm{T}}C^{-1}\mathbf{1}(b\mu_0(\sigma_b^2+b^2\sigma_0^2\mathbf{1}^{\mathrm{T}}C^{-1}\mathbf{1})-b^2\sigma_0^2\mathbf{1}^{\mathrm{T}}C^{-1}(\boldsymbol{Y}-b\boldsymbol{x}+b\mu_0\mathbf{1}))}{\sigma_b^2+\sigma_0^2\mathbf{1}^{\mathrm{T}}C^{-1}\mathbf{1}b^2} \\
= \; & b[x_*-\mathrm{P}_{\boldsymbol{Y}}(X_0)]+\boldsymbol{c}^{\mathrm{T}}C^{-1}(\boldsymbol{Y}-b[\boldsymbol{x}-\mathrm{P}_{\boldsymbol{Y}}(X_0)\mathbf{1}]) \qquad \text{(B.5)}
\end{aligned}
$$

$\mathrm{Cov}([Y(x_*)/\boldsymbol{Y}],[Y(\tilde{x}_*)/\boldsymbol{Y}])$

$$
\begin{aligned}
= \; & \sigma_0^2b^2+\sigma_b^2(\sigma_0^2+(x_*-\mu_0)(\tilde{x}_*-\mu_0))\rho(x_*-\tilde{x}_*)- \\
& (b^2\sigma_0^2\mathbf{1}+\sigma_b^2\boldsymbol{c})^{\mathrm{T}}(b^2\sigma_0^2\mathbf{1}\mathbf{1}^{\mathrm{T}}+\sigma_b^2C)^{-1}(b^2\sigma_0^2\mathbf{1}+\sigma_b^2\tilde{\boldsymbol{c}}) \\
= \; & \sigma_0^2b^2+\sigma_b^2(\sigma_0^2+(x_*-\mu_0)(\tilde{x}_*-\mu_0))\rho(x_*-\tilde{x}_*)- \\
& \frac{b^2\sigma_0^2(b^2\sigma_0^2\mathbf{1}^{\mathrm{T}}C^{-1}\mathbf{1}+\sigma_b^2\mathbf{1}^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{c}}+\sigma_b^2\boldsymbol{c}^{\mathrm{T}}C^{-1}\mathbf{1})}{\sigma_b^2(1+\frac{b^2\sigma_0^2}{\sigma_b^2}\mathbf{1}^{\mathrm{T}}C^{-1}\mathbf{1})}-
\end{aligned}
$$

$$\frac{\sigma_b^4 \boldsymbol{c}^{\mathrm{T}} C^{-1}\tilde{\boldsymbol{c}}}{\sigma_b^2} + \frac{\sigma_b^4 \boldsymbol{c}^{\mathrm{T}} C^{-1}\mathbf{1} b^2\sigma_0^2 \mathbf{1}^{\mathrm{T}} C^{-1}\tilde{\boldsymbol{c}}}{\sigma_b^4(1+\frac{b^2\sigma_0^2}{\sigma_b^2}\mathbf{1}^{\mathrm{T}} C^{-1}\mathbf{1})}$$

$$= \sigma_b^2\Big((\sigma_0^2 + (x_* - \mu_0)(\tilde{x}_* - \mu_0))\rho(x_* - \tilde{x}_*) - \boldsymbol{c}^{\mathrm{T}} C^{-1}\tilde{\boldsymbol{c}} + \frac{(1 - \mathbf{1}^{\mathrm{T}} C^{-1}\tilde{\boldsymbol{c}})b^2\sigma_0^2(1 - \boldsymbol{c}^{\mathrm{T}} C^{-1}\mathbf{1})}{\sigma_b^2 + \sigma_0^2 \mathbf{1}^{\mathrm{T}} C^{-1}\mathbf{1} b^2}\Big) \quad \text{(B.6)}$$

$\mathrm{Cov}([B(x_*)/\boldsymbol{Y}], [Y(\tilde{\boldsymbol{x}}_*)/\boldsymbol{Y}])$

$$= (\tilde{\boldsymbol{x}} - \mu_0)\sigma_b^2\rho(x_* - \tilde{x}_*) - \sigma_b^2 \boldsymbol{d}^{\mathrm{T}}(b^2\sigma_0^2 \mathbf{1}\mathbf{1}^{\mathrm{T}} + \sigma_b^2 C)^{-1}(b^2\sigma_0^2\mathbf{1} + \sigma_b^2\tilde{\boldsymbol{c}})$$

$$= (\tilde{\boldsymbol{x}} - \mu_0)\sigma_b^2\rho(x_* - \tilde{x}_*) - \frac{\sigma_b^2 \boldsymbol{d}^{\mathrm{T}} C^{-1}\sigma_b^2\tilde{\boldsymbol{c}}}{\sigma_b^2} + \frac{\sigma_b^2 \boldsymbol{d}^{\mathrm{T}} C^{-1}\mathbf{1} b^2\sigma_0^2 \mathbf{1}^{\mathrm{T}} C^{-1}\sigma_b^2\tilde{\boldsymbol{c}}}{\sigma_b^4(1+\frac{b^2\sigma_0^2}{\sigma_b^2}\mathbf{1}^{\mathrm{T}} C^{-1}\mathbf{1})} - \frac{\sigma_b^2 b^2\sigma_0^2 d C^{-1}\mathbf{1}}{\sigma_b^2(1+\frac{b^2\sigma_0^2}{\sigma_b^2}\mathbf{1}^{\mathrm{T}} C^{-1}\mathbf{1})}$$

$$= \sigma_b^2\left((\tilde{x} - \mu_0)\rho(x_* - \tilde{x}_*) - \boldsymbol{d}^{\mathrm{T}} C^{-1}\tilde{\boldsymbol{c}} + \frac{\boldsymbol{d}^{\mathrm{T}} C^{-1}\mathbf{1} b^2\sigma_0^2(\mathbf{1}^{\mathrm{T}} C^{-1}\tilde{\boldsymbol{c}} - 1)}{\sigma_b^2 + \sigma_0^2 \mathbf{1}^{\mathrm{T}} C^{-1}\mathbf{1} b^2}\right) \quad \text{(B.7)}$$

B.2 Determinant results

Lemma B.1

$$|M||M_{\overline{ij},\overline{IJ}}| = |M_{\overline{i},\overline{I}}||M_{\overline{j},\overline{J}}| - |M_{\overline{i},\overline{J}}||M_{\overline{j},\overline{I}}|, \qquad i \neq j \; and \; I \neq J$$

where $M_{\overline{a_1 \ldots a_m},\overline{c_1 \ldots c_m}}$ *is the matrix* M *with the* $a_1, \ldots, a_m$ *columns and* $c_1, \ldots, c_m$ *rows removed.*

Proof

If we divide the RHS by the LHS we get

$$Q = \frac{|M_{\overline{i},\overline{I}}|}{|M|} \Big/ \frac{|M_{\overline{ij},\overline{IJ}}|}{|M_{\overline{j},\overline{J}}|} - \frac{|M_{\overline{i},\overline{J}}|}{|M|} \Big/ \frac{|M_{\overline{ij},\overline{IJ}}|}{|M_{\overline{j},\overline{I}}|}$$

by swapping columns and rows we can w.l.o.g. assume $i = I = 1$ and $j = J = 2$ so we have

$$Q = \frac{|M_{\overline{1},\overline{1}}|}{|M|} \Big/ \frac{|M_{\overline{12},\overline{12}}|}{|M_{\overline{2},\overline{2}}|} - \frac{|M_{\overline{1},\overline{2}}|}{|M|} \Big/ \frac{|M_{\overline{12},\overline{12}}|}{|M_{\overline{2},\overline{1}}|}$$

Writing $M = \begin{bmatrix} c_{11} & c_{12} & \boldsymbol{a}_1^{\mathrm{T}} \\ c_{21} & c_{22} & \boldsymbol{a}_2^{\mathrm{T}} \\ \boldsymbol{b}_1 & \boldsymbol{b}_2 & D \end{bmatrix}$ we have

$$Q = \frac{\begin{vmatrix} 1 & c_{12} & \boldsymbol{a}_1^{\mathrm{T}} \\ 0 & c_{22} & \boldsymbol{a}_2^{\mathrm{T}} \\ 0 & \boldsymbol{b}_2 & D \end{vmatrix}}{\begin{vmatrix} c_{11} & c_{12} & \boldsymbol{a}_1^{\mathrm{T}} \\ c_{21} & c_{22} & \boldsymbol{a}_2^{\mathrm{T}} \\ \boldsymbol{b}_1 & \boldsymbol{b}_2 & D \end{vmatrix}} / \frac{\begin{vmatrix} 1 & 0 & \boldsymbol{a}_1^{\mathrm{T}} \\ 0 & 1 & \boldsymbol{a}_2^{\mathrm{T}} \\ 0 & 0 & D \end{vmatrix}}{\begin{vmatrix} c_{11} & 0 & \boldsymbol{a}_1^{\mathrm{T}} \\ c_{21} & 1 & \boldsymbol{a}_2^{\mathrm{T}} \\ \boldsymbol{b}_1 & 0 & D \end{vmatrix}} - \frac{\begin{vmatrix} 0 & c_{12} & \boldsymbol{a}_1^{\mathrm{T}} \\ 1 & c_{22} & \boldsymbol{a}_2^{\mathrm{T}} \\ 0 & \boldsymbol{b}_2 & D \end{vmatrix}}{\begin{vmatrix} c_{11} & c_{12} & \boldsymbol{a}_1^{\mathrm{T}} \\ c_{21} & c_{22} & \boldsymbol{a}_2^{\mathrm{T}} \\ \boldsymbol{b}_1 & \boldsymbol{b}_2 & D \end{vmatrix}} / \frac{\begin{vmatrix} 0 & 1 & \boldsymbol{a}_1^{\mathrm{T}} \\ 1 & 0 & \boldsymbol{a}_2^{\mathrm{T}} \\ 0 & 0 & D \end{vmatrix}}{\begin{vmatrix} c_{11} & 1 & \boldsymbol{a}_1^{\mathrm{T}} \\ c_{21} & 0 & \boldsymbol{a}_2^{\mathrm{T}} \\ \boldsymbol{b}_1 & 0 & D \end{vmatrix}}$$

and applying Cramers rule

$$\begin{aligned} & Q \\ &= \frac{(c_{11} - \boldsymbol{a}_1^{\mathrm{T}} D^{-1} \boldsymbol{b}_1)(c_{22} - \boldsymbol{a}_2^{\mathrm{T}} D^{-1} \boldsymbol{b}_2)}{(c_{11} - \boldsymbol{a}_1^{\mathrm{T}} D^{-1} \boldsymbol{b}_1)(c_{22} - \boldsymbol{a}_2^{\mathrm{T}} D^{-1} \boldsymbol{b}_2) - (c_{12} - \boldsymbol{a}_1^{\mathrm{T}} D^{-1} \boldsymbol{b}_2)(c_{21} - \boldsymbol{a}_2^{\mathrm{T}} D^{-1} \boldsymbol{b}_1)} \\ &\quad - \frac{(c_{12} - \boldsymbol{a}_1^{\mathrm{T}} D^{-1} \boldsymbol{b}_2)(c_{21} - \boldsymbol{a}_2^{\mathrm{T}} D^{-1} \boldsymbol{b}_1)}{(c_{11} - \boldsymbol{a}_1^{\mathrm{T}} D^{-1} \boldsymbol{b}_1)(c_{22} - \boldsymbol{a}_2^{\mathrm{T}} D^{-1} \boldsymbol{b}_2) - (c_{12} - \boldsymbol{a}_1^{\mathrm{T}} D^{-1} \boldsymbol{b}_2)(c_{21} - \boldsymbol{a}_2^{\mathrm{T}} D^{-1} \boldsymbol{b}_1)} \\ &= 1 \end{aligned}$$

So $LHS = RHS$ □

Theorem B.2

$$\lim_{k_1, \ldots, k_m \to \infty} \boldsymbol{f}^{\mathrm{T}} (M + k_1 \boldsymbol{a}_1 \boldsymbol{b}_1^{\mathrm{T}} + \cdots + k_m \boldsymbol{a}_m \boldsymbol{b}_m^{\mathrm{T}})^{-1} \boldsymbol{g} = \frac{\begin{vmatrix} B_m^{\mathrm{T}} M^{-1} A_m & B_m^{\mathrm{T}} M^{-1} \boldsymbol{g} \\ \boldsymbol{f}^{\mathrm{T}} M^{-1} A_m & \boldsymbol{f}^{\mathrm{T}} M^{-1} \boldsymbol{g} \end{vmatrix}}{\left| B_m^{\mathrm{T}} M^{-1} A_m \right|}$$

where $A_q = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_q]$ *and* $B_q = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_q]$

Proof

By induction on m.

$$\begin{aligned}\lim_{k\to\infty} \boldsymbol{f}^{\mathrm{T}}(M+k\boldsymbol{a}\boldsymbol{b}^{\mathrm{T}})^{-1}\boldsymbol{g} &= \lim_{k\to\infty} \boldsymbol{f}^{\mathrm{T}}M^{-1}\boldsymbol{g} - \frac{k\boldsymbol{f}^{\mathrm{T}}M^{-1}\boldsymbol{a}\boldsymbol{b}^{\mathrm{T}}M^{-1}\boldsymbol{g}}{1+k\boldsymbol{b}^{\mathrm{T}}M^{-1}\boldsymbol{a}} \\ &= \boldsymbol{f}^{\mathrm{T}}M^{-1}\boldsymbol{g} - \frac{\boldsymbol{f}^{\mathrm{T}}M^{-1}\boldsymbol{a}\boldsymbol{b}^{\mathrm{T}}M^{-1}\boldsymbol{g}}{\boldsymbol{b}^{\mathrm{T}}M^{-1}\boldsymbol{a}} \\ &= \frac{\begin{vmatrix} \boldsymbol{b}^{\mathrm{T}}M^{-1}\boldsymbol{a} & \boldsymbol{b}^{\mathrm{T}}M^{-1}\boldsymbol{g} \\ \boldsymbol{f}^{\mathrm{T}}M^{-1}\boldsymbol{a} & \boldsymbol{f}^{\mathrm{T}}M^{-1}\boldsymbol{g} \end{vmatrix}}{\left|\boldsymbol{b}^{\mathrm{T}}M^{-1}\boldsymbol{a}\right|}\end{aligned}$$

So true for $m = 1$, assume true for $m = p$;

$$\begin{aligned}&\lim_{k_1,\ldots,k_p,k_{p+1}\to\infty} \boldsymbol{f}^{\mathrm{T}}(M+k_1\boldsymbol{a}_1\boldsymbol{b}_1{}^{\mathrm{T}}+\cdots+k_{p+1}\boldsymbol{a}_{p+1}\boldsymbol{b}_{p+1}{}^{\mathrm{T}})^{-1}\boldsymbol{g} \\ &= \boldsymbol{f}^{\mathrm{T}}M_*^{-1}\boldsymbol{g} - \frac{\boldsymbol{f}^{\mathrm{T}}M_*^{-1}\boldsymbol{a}_{p+1}\boldsymbol{b}_{p+1}{}^{\mathrm{T}}M_*^{-1}\boldsymbol{g}}{\boldsymbol{b}_{p+1}{}^{\mathrm{T}}M_*^{-1}\boldsymbol{a}_{p+1}} \\ &= \frac{\begin{vmatrix} B_p^{\mathrm{T}}M^{-1}A_p & B_p^{\mathrm{T}}M^{-1}\boldsymbol{g} \\ \boldsymbol{f}^{\mathrm{T}}M^{-1}A_p & \boldsymbol{f}^{\mathrm{T}}M^{-1}\boldsymbol{g} \end{vmatrix}}{\left|B_p^{\mathrm{T}}M^{-1}A_p\right|} \\ &\qquad \frac{\begin{vmatrix} B_p^{\mathrm{T}}M^{-1}A_p & B_p^{\mathrm{T}}M^{-1}\boldsymbol{a}_{p+1} \\ \boldsymbol{f}^{\mathrm{T}}M^{-1}A_p & \boldsymbol{f}^{\mathrm{T}}M^{-1}\boldsymbol{a}_{p+1} \end{vmatrix}}{\left|B_p^{\mathrm{T}}M^{-1}A_p\right|} \frac{\begin{vmatrix} B_p^{\mathrm{T}}M^{-1}A_p & B_p^{\mathrm{T}}M^{-1}\boldsymbol{g} \\ \boldsymbol{b}_{p+1}{}^{\mathrm{T}}M^{-1}A_p & \boldsymbol{b}_{p+1}{}^{\mathrm{T}}M^{-1}\boldsymbol{g} \end{vmatrix}}{\left|B_p^{\mathrm{T}}M^{-1}A_p\right|} \\ &\qquad \frac{\begin{vmatrix} B_p^{\mathrm{T}}M^{-1}A_p & B_p^{\mathrm{T}}M^{-1}\boldsymbol{a}_{p+1} \\ \boldsymbol{b}_{p+1}{}^{\mathrm{T}}M^{-1}A_p & \boldsymbol{b}_{p+1}{}^{\mathrm{T}}M^{-1}\boldsymbol{a}_{p+1} \end{vmatrix}}{\left|B_p^{\mathrm{T}}M^{-1}A_p\right|} \\ &= \frac{\begin{vmatrix} B_p^{\mathrm{T}}M^{-1}A_p & B_p^{\mathrm{T}}M^{-1}\boldsymbol{a}_{p+1} & B_p^{\mathrm{T}}M^{-1}\boldsymbol{g} \\ \boldsymbol{b}_{p+1}{}^{\mathrm{T}}M^{-1}A_p & \boldsymbol{b}_{p+1}{}^{\mathrm{T}}M^{-1}\boldsymbol{a}_{p+1} & \boldsymbol{b}_{p+1}{}^{\mathrm{T}}M^{-1}\boldsymbol{g} \\ \boldsymbol{f}^{\mathrm{T}}M^{-1}A_p & \boldsymbol{b}_{p+1}{}^{\mathrm{T}}M^{-1}\boldsymbol{a}_{p+1} & \boldsymbol{f}^{\mathrm{T}}M^{-1}\boldsymbol{g} \end{vmatrix}}{\begin{vmatrix} B_p^{\mathrm{T}}M^{-1}A_p & B_p^{\mathrm{T}}M^{-1}\boldsymbol{a}_{p+1} \\ \boldsymbol{b}_{p+1}{}^{\mathrm{T}}M^{-1}A_p & \boldsymbol{b}_{p+1}{}^{\mathrm{T}}M^{-1}\boldsymbol{a}_{p+1} \end{vmatrix}}\end{aligned}$$

$$= \frac{\begin{vmatrix} B_{p+1}^{\mathrm{T}} M^{-1} A_{p+1} & B_{p+1}^{\mathrm{T}} M^{-1} \boldsymbol{g} \\ \boldsymbol{f}^{\mathrm{T}} M^{-1} A_{p+1} & \boldsymbol{f}^{\mathrm{T}} M^{-1} \boldsymbol{g} \end{vmatrix}}{\left| B_{p+1}^{\mathrm{T}} M^{-1} A_{p+1} \right|}$$

(where $M_* = \lim_{k_1,\ldots,k_p}(M + k_1 \boldsymbol{a}_1 \boldsymbol{b}_1{}^{\mathrm{T}} + \cdots + k_p \boldsymbol{a}_p \boldsymbol{b}_p{}^{\mathrm{T}})$) So if true for $m = p$ then also true for $m = p + 1$. So by inductive argument true for all m. □

Theorem B.3

$$\lim_{k_1,\ldots,k_m \to \infty} k_m \boldsymbol{b}_m{}^{\mathrm{T}} (M + k_1 \boldsymbol{a}_1 \boldsymbol{b}_1^{\mathrm{T}} + \cdots + k_m \boldsymbol{a}_m \boldsymbol{b}_m^{\mathrm{T}})^{-1} \boldsymbol{g} = \frac{\left| B_m^{\mathrm{T}} M^{-1} A_{m-1} \; B_m^{\mathrm{T}} M^{-1} \boldsymbol{g} \right|}{\left| B_m^{\mathrm{T}} M^{-1} A_m \right|}$$

Proof

$$\begin{aligned}
&\lim_{k_1,\ldots,k_m \to \infty} k_m \boldsymbol{b}_m{}^{\mathrm{T}} (M + k_1 \boldsymbol{a}_1 \boldsymbol{b}_1^{\mathrm{T}} + \cdots + k_m \boldsymbol{a}_m \boldsymbol{b}_m^{\mathrm{T}})^{-1} \boldsymbol{g} \\
&= \lim_{k_m \to \infty} k_m \boldsymbol{b}_m{}^{\mathrm{T}} (M_* + k_m \boldsymbol{a}_m \boldsymbol{b}_m{}^{\mathrm{T}})^{-1} \boldsymbol{g} \\
&= \lim_{k_m \to \infty} k_m \boldsymbol{b}_m{}^{\mathrm{T}} M_*^{-1} \boldsymbol{g} - \frac{k_m^2 \boldsymbol{b}_m{}^{\mathrm{T}} M_*^{-1} \boldsymbol{a}_m \boldsymbol{b}_m{}^{\mathrm{T}} M_*^{-1} \boldsymbol{g}}{1 + k_m \boldsymbol{b}_m{}^{\mathrm{T}} M_*^{-1} \boldsymbol{a}_m} \\
&= \lim_{k_m \to \infty} \frac{k_m \boldsymbol{b}_m{}^{\mathrm{T}} M_*^{-1} \boldsymbol{g}}{1 + k_m \boldsymbol{b}_m{}^{\mathrm{T}} M_*^{-1} \boldsymbol{a}_m} \\
&= \frac{\boldsymbol{b}_m{}^{\mathrm{T}} M_*^{-1} \boldsymbol{g}}{\boldsymbol{b}_m{}^{\mathrm{T}} M_*^{-1} \boldsymbol{a}_m} \\
&= \frac{\begin{vmatrix} B_m^{\mathrm{T}} M^{-1} A_{m-1} & B_m^{\mathrm{T}} M^{-1} \boldsymbol{g} \end{vmatrix}}{\left| B_m^{\mathrm{T}} M^{-1} A_m \right|}
\end{aligned}$$

where $M_* = \lim_{k_1,\ldots,k_{m-1} \to \infty}(M + k_1 \boldsymbol{a}_1 \boldsymbol{b}_1{}^{\mathrm{T}} + \cdots + k_{m-1} \boldsymbol{a}_{m-1} \boldsymbol{b}_{m-1}{}^{\mathrm{T}})$ □

Theorem B.4

$$\lim_{k_1,\ldots,k_m\to\infty} k_m \boldsymbol{b}_m^{\mathrm{T}}(M + k_1\boldsymbol{a}_1\boldsymbol{b}_1^{\mathrm{T}} + \cdots + k_m\boldsymbol{a}_m\boldsymbol{b}_m^{\mathrm{T}})^{-1}\boldsymbol{a}_{m-1}k_{m-1} = \frac{\begin{vmatrix} B_{m-2}^{\mathrm{T}}M^{-1}A_{m-1} \\ \boldsymbol{b}_m^{\mathrm{T}}M^{-1}A_{m-1} \end{vmatrix}}{\left|B_m^{\mathrm{T}}M^{-1}A_m\right|}$$

Proof

$$\begin{aligned}
&\lim_{k_1,\ldots,k_m\to\infty} k_m \boldsymbol{b}_m^{\mathrm{T}}(M + k_1\boldsymbol{a}_1\boldsymbol{b}_1^{\mathrm{T}} + \cdots + k_m\boldsymbol{a}_m\boldsymbol{b}_m^{\mathrm{T}})^{-1}\boldsymbol{a}_{m-1}k_{m-1} \\
&= \lim_{k_{m-1},k_m\to\infty} k_m \boldsymbol{b}_m^{\mathrm{T}}(M_* + k_{m-1}\boldsymbol{a}_{m-1}\boldsymbol{b}_{m-1}^{\mathrm{T}} + k_m\boldsymbol{a}_m\boldsymbol{b}_m^{\mathrm{T}})^{-1}\boldsymbol{a}_{m-1}k_{m-1} \\
&= \lim_{k_{m-1}\to\infty} \frac{\boldsymbol{b}_m^{\mathrm{T}}(M_* + k_{m-1}\boldsymbol{a}_{m-1}\boldsymbol{b}_{m-1}^{\mathrm{T}})^{-1}\boldsymbol{a}_{m-1}k_{m-1}}{\boldsymbol{b}_m^{\mathrm{T}}(M_* + k_{m-1}\boldsymbol{a}_{m-1}\boldsymbol{b}_{m-1}^{\mathrm{T}})^{-1}\boldsymbol{a}_m} \\
&= \frac{\begin{vmatrix} B_{m-2}^{\mathrm{T}}M^{-1}A_{m-1} \\ \boldsymbol{b}_m^{\mathrm{T}}M^{-1}A_{m-1} \end{vmatrix}}{\left|B_{m-1}^{\mathrm{T}}M^{-1}A_{m-1}\right|} \Big/ \frac{\left|B_m^{\mathrm{T}}M^{-1}A_m\right|}{\left|B_{m-1}^{\mathrm{T}}M^{-1}A_{m-1}\right|} \\
&= \frac{\begin{vmatrix} B_{m-2}^{\mathrm{T}}M^{-1}A_{m-1} \\ \boldsymbol{b}_m^{\mathrm{T}}M^{-1}A_{m-1} \end{vmatrix}}{\left|B_m^{\mathrm{T}}M^{-1}A_m\right|}
\end{aligned}$$

where $M_* = \lim_{k_1,\ldots,k_{m-2}\to\infty}(M + k_1\boldsymbol{a}_1\boldsymbol{b}_1^{\mathrm{T}} + \cdots + k_{m-2}\boldsymbol{a}_{m-2}\boldsymbol{b}_{m-2}^{\mathrm{T}})$ □

Theorem B.5

$$\lim_{k_1,\ldots,k_m\to\infty} k_m - k_m \boldsymbol{b}_m^{\mathrm{T}}(M + k_1\boldsymbol{a}_1\boldsymbol{b}_1^{\mathrm{T}} + \cdots + k_m\boldsymbol{a}_m\boldsymbol{b}_m^{\mathrm{T}})^{-1}\boldsymbol{a}_m k_m = \frac{\left|B_{m-1}^{\mathrm{T}}M^{-1}A_{m-1}\right|}{\left|B_m^{\mathrm{T}}M^{-1}A_m\right|}$$

Proof

$$
\begin{aligned}
&\lim_{k_1,\ldots,k_m\to\infty} k_m - k_m \boldsymbol{b}_m^{\mathrm{T}}(M + k_1\boldsymbol{a}_1\boldsymbol{b}_1^{\mathrm{T}} + \cdots + k_m\boldsymbol{a}_m\boldsymbol{b}_m^{\mathrm{T}})^{-1}\boldsymbol{a}_m k_m \\
&= \lim_{k_m\to\infty} k_m - k_m^2 \boldsymbol{b}_m{}^{\mathrm{T}}(M_* + k_m\boldsymbol{a}_m\boldsymbol{b}_m{}^{\mathrm{T}})^{-1}\boldsymbol{a}_m \\
&= \lim_{k_m\to\infty} k_m - k_m^2 \boldsymbol{b}_m{}^{\mathrm{T}} M_*^{-1}\boldsymbol{a}_m - \frac{k_m^3(\boldsymbol{b}_m{}^{\mathrm{T}} M_*^{-1}\boldsymbol{a}_m)^2}{1 + k_m\boldsymbol{b}_m{}^{\mathrm{T}} M_*^{-1}\boldsymbol{a}_m} \\
&= \lim_{k_m\to\infty} \frac{k_m}{1 + k_m\boldsymbol{b}_m{}^{\mathrm{T}} M_*^{-1}\boldsymbol{a}_m} \\
&= \frac{1}{\boldsymbol{b}_m{}^{\mathrm{T}} M_*^{-1}\boldsymbol{a}_m} \\
&= \frac{\left|B_{m-1}^{\mathrm{T}} M^{-1} A_{m-1}\right|}{\left|B_m^{\mathrm{T}} M^{-1} A_m\right|}
\end{aligned}
$$

where $M_* = \lim_{k_1,\ldots,k_{m-1}\to\infty}(M + k_1\boldsymbol{a}_1\boldsymbol{b}_1{}^{\mathrm{T}} + \cdots + k_{m-1}\boldsymbol{a}_{m-1}\boldsymbol{b}_{m-1}{}^{\mathrm{T}})$ □

B.3 Results for "ignorance prior" – Section 5.5

$$
\begin{aligned}
&\mathrm{P}\boldsymbol{y}(Y(x_*)) \\
&= b(x_* - \mu_0) + \lim_{\sigma_M\to\infty} \left(\sigma_M^2(x_* - \mu_0)(\boldsymbol{x} - \mu_0\mathbf{1})^{\mathrm{T}} + (b^2 + \sigma_M^2)\sigma_0^2\mathbf{1}^{\mathrm{T}} + \sigma_b^2\boldsymbol{c}^{\mathrm{T}}\right) \\
&\quad \left(\sigma_M^2(\boldsymbol{x} - \mu_0\mathbf{1})(\boldsymbol{x} - \mu_0\mathbf{1})^{\mathrm{T}} + (b^2 + \sigma_M^2)\sigma_0^2\mathbf{1}\mathbf{1}^{\mathrm{T}} + \sigma_b^2 C\right)^{-1}(\boldsymbol{Y} - b(\boldsymbol{x} - \mu_0\mathbf{1})) \\
&= b(x_* - \mu_0) + \left|\begin{pmatrix} \mathbf{1}^{\mathrm{T}} \\ (\boldsymbol{x} - \mu_0\mathbf{1})^{\mathrm{T}} \end{pmatrix} \frac{C^{-1}}{\sigma_b^2}\begin{pmatrix} 1 & (\boldsymbol{x} - \mu_0\mathbf{1}) \end{pmatrix}\right|^{-1} \\
&\quad \begin{pmatrix} (x_* - \mu_0)\left|\begin{pmatrix} \mathbf{1}^{\mathrm{T}} \\ (\boldsymbol{x} - \mu_0\mathbf{1})^{\mathrm{T}} \end{pmatrix} \frac{C^{-1}}{\sigma_b^2}\begin{pmatrix} 1 & (\boldsymbol{Y} - b(\boldsymbol{x} - \mu_0\mathbf{1})) \end{pmatrix}\right| \\ + \left|\begin{pmatrix} \mathbf{1}^{\mathrm{T}} \\ (\boldsymbol{x} - \mu_0\mathbf{1})^{\mathrm{T}} \\ \sigma_b^2\boldsymbol{c}^{\mathrm{T}} \end{pmatrix} \frac{C^{-1}}{\sigma_b^2}\begin{pmatrix} 1 & (\boldsymbol{x} - \mu_0\mathbf{1}) & (\boldsymbol{Y} - b(\boldsymbol{x} - \mu_0\mathbf{1})) \end{pmatrix}\right|
\end{aligned}
$$

$$
\left|\begin{pmatrix} (\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} \\ \mathbf{1}^{\mathrm{T}} \end{pmatrix} \frac{C^{-1}}{\sigma_b^2} \begin{pmatrix} (\boldsymbol{x}-\mu_0\mathbf{1}) & (\boldsymbol{Y}-b(\boldsymbol{x}-\mu_0\mathbf{1})) \end{pmatrix}\right|\Bigg)
$$

$$
\begin{aligned}
= \; & \left|F^{\mathrm{T}}C^{-1}F\right|^{-1}\Bigg(b(x_*-\mu_0)\left|F^{\mathrm{T}}C^{-1}F\right| + \frac{1}{\sigma_b^2}\left|\begin{pmatrix} F^{\mathrm{T}} \\ \sigma_b^2\boldsymbol{c}^{\mathrm{T}} \end{pmatrix} C^{-1}\begin{pmatrix} F & \boldsymbol{Y} \end{pmatrix}\right| \\
& +\mu_0\left|F^{\mathrm{T}}C^{-1}\begin{pmatrix} \mathbf{1} & \boldsymbol{Y} \end{pmatrix}\right| - b(x_*-\mu_0)\left|F^{\mathrm{T}}C^{-1}F\right| \\
& -\left|F^{\mathrm{T}}C^{-1}\begin{pmatrix} \boldsymbol{x} & \boldsymbol{Y} \end{pmatrix}\right| + (x_*-\mu_0)\left|F^{\mathrm{T}}C^{-1}\begin{pmatrix} \mathbf{1} & \boldsymbol{Y} \end{pmatrix}\right|\Bigg) \\
= \; & \frac{\begin{vmatrix} F^{\mathrm{T}}C^{-1}F & F^{\mathrm{T}}C^{-1}\boldsymbol{Y} \\ \boldsymbol{c}^{\mathrm{T}}C^{-1}F - \boldsymbol{f}^{\mathrm{T}} & \boldsymbol{c}^{\mathrm{T}}C^{-1}\boldsymbol{Y} \end{vmatrix}}{\left|F^{\mathrm{T}}C^{-1}F\right|} \\
= \; & \boldsymbol{c}^{\mathrm{T}}C^{-1}\boldsymbol{Y} - (\boldsymbol{c}^{\mathrm{T}}C^{-1}F - \boldsymbol{f}^{\mathrm{T}})(F^{\mathrm{T}}C^{-1}F)^{-1}F^{\mathrm{T}}C^{-1}\boldsymbol{Y}
\end{aligned} \tag{B.8}
$$

$$
\begin{aligned}
& \mathrm{Cov}([Y(x_*)/\boldsymbol{Y}],[Y(\tilde{x}_*)/\boldsymbol{Y}]) \\
= \; & \lim_{\sigma_M\to\infty} \sigma_M^2(x_*-\mu_0)(\tilde{x}_*-\mu_0) + (b^2+\sigma_M^2)\sigma_0^2 + \sigma_b^2 c(x_*,x_*) - \\
& \left(\sigma_M^2(x_*-\mu_0)(\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} + (b^2+\sigma_M^2)\sigma_0^2\mathbf{1}^{\mathrm{T}} + \sigma_b^2\boldsymbol{c}^{\mathrm{T}}\right) \\
& \quad \left(\sigma_M^2(\boldsymbol{x}-\mu_0\mathbf{1})(\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} + (b^2+\sigma_M^2)\sigma_0^2\mathbf{1}\mathbf{1}^{\mathrm{T}} + \sigma_b^2 C\right)^{-1} \\
& \quad\quad \left(\sigma_M^2(\tilde{x}_*-\mu_0)(\boldsymbol{x}-\mu_0\mathbf{1}) + (b^2+\sigma_M^2)\sigma_0^2\mathbf{1} + \sigma_b^2\tilde{\boldsymbol{c}}\right) \\
= \; & \sigma_b^2 c(x_*,x_*) - \left|\begin{pmatrix} \mathbf{1}^{\mathrm{T}} \\ (\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} \end{pmatrix} \frac{C^{-1}}{\sigma_b^2}\begin{pmatrix} \mathbf{1} & (\boldsymbol{x}-\mu_0\mathbf{1}) \end{pmatrix}\right|^{-1} \\
& \left(\left|\begin{pmatrix} \mathbf{1}^{\mathrm{T}} \\ (\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} \\ \sigma_b^2\boldsymbol{c}^{\mathrm{T}} \end{pmatrix} \frac{C^{-1}}{\sigma_b^2}\begin{pmatrix} \mathbf{1} & (\boldsymbol{x}-\mu_0\mathbf{1}) & \sigma_b^2\tilde{\boldsymbol{c}} \end{pmatrix}\right|\right. \\
& -(x_*-\mu_0)(\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}}\frac{C^{-1}}{\sigma_b^2}\mathbf{1} - (\tilde{x}_*-\mu_0)\mathbf{1}^{\mathrm{T}}\frac{C^{-1}}{\sigma_b^2}(\boldsymbol{x}-\mu_0\mathbf{1}) \\
& +(x_*-\mu_0)(\tilde{x}_*-\mu_0)\mathbf{1}^{\mathrm{T}}\frac{C^{-1}}{\sigma_b^2}\mathbf{1} + (\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}}\frac{C^{-1}}{\sigma_b^2}(\boldsymbol{x}-\mu_0\mathbf{1}) \\
& \left.-(x_*-\mu_0)\left|\begin{pmatrix} \mathbf{1}^{\mathrm{T}} \\ (\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} \end{pmatrix} \frac{C^{-1}}{\sigma_b^2}\begin{pmatrix} \mathbf{1} & \sigma_b^2\tilde{\boldsymbol{c}} \end{pmatrix}\right|\right)
\end{aligned}
$$

$$
\begin{aligned}
&-(\tilde{x}_* - \mu_0)\left|\begin{pmatrix} \mathbf{1}^{\mathrm{T}} \\ \sigma_b^2 \boldsymbol{c}^{\mathrm{T}} \end{pmatrix} \frac{C^{-1}}{\sigma_b^2} \begin{pmatrix} \mathbf{1} & (\boldsymbol{x}-\mu_0\mathbf{1}) \end{pmatrix}\right| \\
&-\left|\begin{pmatrix} (\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} \\ \mathbf{1}^{\mathrm{T}} \end{pmatrix} \frac{C^{-1}}{\sigma_b^2} \begin{pmatrix} (\boldsymbol{x}-\mu_0\mathbf{1}) & \sigma_b^2\tilde{\boldsymbol{c}} \end{pmatrix}\right| \\
&-\left|\begin{pmatrix} (\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} \\ \sigma_b^2 \boldsymbol{c}^{\mathrm{T}} \end{pmatrix} \frac{C^{-1}}{\sigma_b^2} \begin{pmatrix} (\boldsymbol{x}-\mu_0\mathbf{1}) & \mathbf{1} \end{pmatrix}\right|\Bigg) \\
= \; & \sigma_b^2 c(x_*, x_*) - \left|F^{\mathrm{T}} C^{-1} F\right|^{-1} \left(\frac{1}{\sigma_b^2}\left|\begin{pmatrix} F^{\mathrm{T}} \\ \sigma_b^2 \boldsymbol{c}^{\mathrm{T}} \end{pmatrix} C^{-1} \begin{pmatrix} F & \sigma_b^2\tilde{\boldsymbol{c}} \end{pmatrix}\right|\right. \\
&+\sigma_b^2(\boldsymbol{x} - \tilde{x}_*\mathbf{1})^{\mathrm{T}} C^{-1}(\boldsymbol{x} - x_*\mathbf{1}) + \left|F^{\mathrm{T}} C^{-1} \begin{pmatrix} (\boldsymbol{x}-\mu_0\mathbf{1}) & \sigma_b^2\tilde{\boldsymbol{c}} \end{pmatrix}\right| \\
&-(x_* - \mu_0)\left|F^{\mathrm{T}} C^{-1} \begin{pmatrix} \mathbf{1} & \sigma_b^2\tilde{\boldsymbol{c}} \end{pmatrix}\right| + \left|\begin{pmatrix} (\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} \\ \sigma_b^2 \boldsymbol{c}^{\mathrm{T}} \end{pmatrix} C^{-1} F\right| \\
&-\mu_0\left|F^{\mathrm{T}} C^{-1} \begin{pmatrix} \mathbf{1} & \sigma_b^2\tilde{\boldsymbol{c}} \end{pmatrix}\right| - (\tilde{x}_* - \mu_0)\left|\begin{pmatrix} \mathbf{1}^{\mathrm{T}} \\ \sigma_b^2 \boldsymbol{c}^{\mathrm{T}} \end{pmatrix} C^{-1} F\right| \\
&\left.-\mu_0\left|\begin{pmatrix} \mathbf{1}^{\mathrm{T}} \\ \sigma_b^2 \boldsymbol{c}^{\mathrm{T}} \end{pmatrix} C^{-1} F\right|\right) \\
= \; & \sigma_b^2 c(x_*, x_*) - \sigma_b^2 \frac{\begin{vmatrix} F^{\mathrm{T}} C^{-1} F & F^{\mathrm{T}} C^{-1}\tilde{\boldsymbol{c}} - \tilde{\boldsymbol{f}} \\ \boldsymbol{c}^{\mathrm{T}} C^{-1} F - \boldsymbol{f}^{\mathrm{T}} & \boldsymbol{c}^{\mathrm{T}} C^{-1}\tilde{\boldsymbol{c}} \end{vmatrix}}{\left|F^{\mathrm{T}} C^{-1} F\right|} \\
= \; & \sigma_b^2 \left(c(x_*, x_*) + (\boldsymbol{c}^{\mathrm{T}} C^{-1} F - \boldsymbol{f}^{\mathrm{T}})(F^{\mathrm{T}} C^{-1} F)^{-1}(F^{\mathrm{T}} C^{-1}\tilde{\boldsymbol{c}} - \tilde{\boldsymbol{f}}) - \boldsymbol{c}^{\mathrm{T}} C^{-1}\tilde{\boldsymbol{c}}\right) \qquad \text{(B.9)}
\end{aligned}
$$

$$
\begin{aligned}
&\mathrm{P}_{\boldsymbol{Y}}(B) \\
= \; & b + \lim_{\sigma_M \to \infty} \sigma_M^2 (\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} \\
&\left(\sigma_M^2 (\boldsymbol{x}-\mu_0\mathbf{1})(\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} + (b^2 + \sigma_M^2)\sigma_0^2 \mathbf{1}\mathbf{1}^{\mathrm{T}} + \sigma_b^2 C\right)^{-1} (\boldsymbol{Y} - b(\boldsymbol{x}-\mu_0\mathbf{1})) \\
= \; & b + \frac{\left|\begin{pmatrix} \mathbf{1}^{\mathrm{T}} \\ (\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} \end{pmatrix} \frac{C^{-1}}{\sigma_b^2} \begin{pmatrix} \mathbf{1} & (\boldsymbol{Y} - b(\boldsymbol{x}-\mu_0\mathbf{1})) \end{pmatrix}\right|}{\left|\begin{pmatrix} \mathbf{1}^{\mathrm{T}} \\ (\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} \end{pmatrix} \frac{C^{-1}}{\sigma_b^2} \begin{pmatrix} \mathbf{1} & (\boldsymbol{x}-\mu_0\mathbf{1}) \end{pmatrix}\right|}
\end{aligned}
$$

$$
\begin{aligned}
&= b + \frac{\left|F^{\mathrm{T}}C^{-1}(\ \mathbf{1}\ \ \boldsymbol{Y}\ )\right|}{|F^{\mathrm{T}}C^{-1}F|} - b\frac{|F^{\mathrm{T}}C^{-1}F|}{|F^{\mathrm{T}}C^{-1}F|} \\
&= \frac{\left|F^{\mathrm{T}}C^{-1}(\ \mathbf{1}\ \ \boldsymbol{Y}\ )\right|}{|F^{\mathrm{T}}C^{-1}F|} \\
&= (\ 0\ \ 1\ )(F^{\mathrm{T}}C^{-1}F)^{-1}(F^{\mathrm{T}}C^{-1}\boldsymbol{y}) \qquad \text{(B.10)}
\end{aligned}
$$

$\mathrm{Var}[B/\boldsymbol{Y}]$

$$
\begin{aligned}
&= \lim_{\sigma_M \to \infty} \sigma_M^2 - \sigma_M^2(\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} \\
&\quad \left(\sigma_M^2(\boldsymbol{x}-\mu_0\mathbf{1})(\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} + (b^2+\sigma_M^2)\sigma_0^2\mathbf{1}\mathbf{1}^{\mathrm{T}} + \sigma_b^2 C\right)^{-1} \sigma_M^2(\boldsymbol{x}-\mu_0\mathbf{1}) \\
&= \frac{|\mathbf{1}^{\mathrm{T}}\frac{C^{-1}}{\sigma_b^2}\mathbf{1}|}{\left|\begin{pmatrix} \mathbf{1}^{\mathrm{T}} \\ (\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} \end{pmatrix} \frac{C^{-1}}{\sigma_b^2}(\ \mathbf{1}\ \ (\boldsymbol{x}-\mu_0\mathbf{1})\ )\right|} \\
&= \frac{\sigma_b^2|\mathbf{1}^{\mathrm{T}}C^{-1}\mathbf{1}|}{|F^{\mathrm{T}}C^{-1}F|} \\
&= \sigma_b^2\left[\boldsymbol{x}^{\mathrm{T}}C^{-1}\boldsymbol{x} - \frac{(\boldsymbol{x}^{\mathrm{T}}C^{-1}\mathbf{1})^2}{\mathbf{1}^{\mathrm{T}}C^{-1}\mathbf{1}}\right]^{-1} \qquad \text{(B.11)}
\end{aligned}
$$

$\mathrm{Cov}([B/\boldsymbol{Y}],[Y(\tilde{x}_*)/\boldsymbol{Y}])$

$$
\begin{aligned}
&= \lim_{\sigma_M \to \infty} \sigma_M^2(\tilde{x}_* - \mu_0) - \sigma_M^2(\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} \\
&\qquad \left(\sigma_M^2(\boldsymbol{x}-\mu_0\mathbf{1})(\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} + (b^2+\sigma_M^2)\sigma_0^2\mathbf{1}\mathbf{1}^{\mathrm{T}} + \sigma_b^2 C\right)^{-1} \\
&\qquad\quad \left(\sigma_M^2(\tilde{x}_* - \mu_0)(\boldsymbol{x}-\mu_0\mathbf{1}) + (b^2+\sigma_M^2)\sigma_0^2\mathbf{1} + \sigma_b^2\tilde{\boldsymbol{c}}\right) \\
&= \left|\begin{pmatrix} \mathbf{1}^{\mathrm{T}} \\ (\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} \end{pmatrix} \frac{C^{-1}}{\sigma_b^2}(\ \mathbf{1}\ \ (\boldsymbol{x}-\mu_0\mathbf{1})\ )\right|^{-1} \\
&\quad \left((\tilde{x}_* - \mu_0)|\mathbf{1}^{\mathrm{T}}\frac{C^{-1}}{\sigma_b^2}\mathbf{1}| - |(\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}}\frac{C^{-1}}{\sigma_b^2}\mathbf{1}| - \left|\begin{pmatrix} \mathbf{1}^{\mathrm{T}} \\ (\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} \end{pmatrix} \frac{C^{-1}}{\sigma_b^2}\left(1\ \ \sigma_b^2\tilde{\boldsymbol{c}}\right)\right|\right) \\
&= \sigma_b^2\left(\frac{\left|F^{\mathrm{T}}C^{-1}(1\ \ \tilde{\boldsymbol{c}})\right|}{\left|F^{\mathrm{T}}C^{-1}F\right|} - \tilde{x}_*\frac{\left|\mathbf{1}^{\mathrm{T}}C^{-1}\mathbf{1}\right|}{\left|F^{\mathrm{T}}C^{-1}F\right|} - \frac{\left|\boldsymbol{x}^{\mathrm{T}}C^{-1}\mathbf{1}\right|}{\left|F^{\mathrm{T}}C^{-1}F\right|}\right) \\
&= \frac{\sigma_b^2\left|F^{\mathrm{T}}C^{-1}\mathbf{1} \quad F^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{c}} - \tilde{\boldsymbol{f}})\right|}{\left|F^{\mathrm{T}}C^{-1}F\right|} \\
&= (0\ \ 1)(F^{\mathrm{T}}C^{-1}F)^{-1}(F^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{c}} - \tilde{\boldsymbol{f}}) \qquad \text{(B.12)}
\end{aligned}
$$

$$
\begin{aligned}
&\mathrm{P}_{\boldsymbol{Y}}(B(x_*)) \\
&= b + \lim_{\sigma_M \to \infty} (\sigma_M^2(\boldsymbol{x}-\mu_0\mathbf{1}) + \sigma_b \boldsymbol{d})^{\mathrm{T}} \\
&\quad \left(\sigma_M^2(\boldsymbol{x}-\mu_0\mathbf{1})(\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} + (b^2+\sigma_M^2)\sigma_0^2\mathbf{1}\mathbf{1}^{\mathrm{T}} + \sigma_b^2 C\right)^{-1} (\boldsymbol{Y} - b(\boldsymbol{x}-\mu_0\mathbf{1})) \\
&= \frac{\left|F^{\mathrm{T}} C^{-1}(\ \mathbf{1}\quad \boldsymbol{Y}\ )\right|}{|F^{\mathrm{T}}\frac{C^{-1}}{\sigma_b^2}F|} + \\
&\qquad \frac{\left|\begin{pmatrix} \mathbf{1}^{\mathrm{T}} \\ (\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} \\ \sigma_b^2 \boldsymbol{d}^{\mathrm{T}} \end{pmatrix} \frac{C^{-1}}{\sigma_b^2}(\ \mathbf{1}\quad (\boldsymbol{x}-\mu_0\mathbf{1})\quad (\boldsymbol{Y}-b(\boldsymbol{x}-\mu_0\mathbf{1}))\ )\right|}{\left|\begin{pmatrix} \mathbf{1}^{\mathrm{T}} \\ (\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} \end{pmatrix} \frac{C^{-1}}{\sigma_b^2}(\ \mathbf{1}\quad (\boldsymbol{x}-\mu_0\mathbf{1})\ )\right|} \\
&= \frac{\left|F^{\mathrm{T}}\frac{C^{-1}}{\sigma_b^2}(\ \mathbf{1}\quad \boldsymbol{Y}\ )\right| + \frac{1}{\sigma_b^2}\left|\begin{pmatrix} F^{\mathrm{T}} \\ \sigma_b^2 \boldsymbol{d}^{\mathrm{T}} \end{pmatrix} C^{-1}(\ F\quad \boldsymbol{Y}\ )\right|}{|F^{\mathrm{T}}C^{-1}F|} \\
&= \frac{\begin{vmatrix} F^{\mathrm{T}}C^{-1}F & F^{\mathrm{T}}C^{-1}\boldsymbol{Y} \\ \boldsymbol{d}^{\mathrm{T}}C^{-1}F - (0\ 1) & \boldsymbol{d}^{\mathrm{T}}C^{-1}\boldsymbol{Y} \end{vmatrix}}{|F^{\mathrm{T}}C^{-1}F|} \\
&= \boldsymbol{d}^{\mathrm{T}}C^{-1}\boldsymbol{Y} - (\boldsymbol{d}^{\mathrm{T}}C^{-1}F - (0\ 1))(F^{\mathrm{T}}C^{-1}F)^{-1}F^{\mathrm{T}}C^{-1}\boldsymbol{Y} \qquad \text{(B.13)}
\end{aligned}
$$

$$
\begin{aligned}
&\mathrm{Cov}([B(x_*)/\boldsymbol{Y}],[B(\tilde{x}_*)/\boldsymbol{Y}]) \\
&= \lim_{\sigma_M \to \infty} \sigma_M^2 + \sigma_b^2\rho(x_*-\tilde{x}_*) - (\sigma_M^2(\boldsymbol{x}-\mu_0\mathbf{1}) + \sigma_b\boldsymbol{d})^{\mathrm{T}} \\
&\qquad \left(\sigma_M^2(\boldsymbol{x}-\mu_0\mathbf{1})(\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} + (b^2+\sigma_M^2)\sigma_0^2\mathbf{1}\mathbf{1}^{\mathrm{T}} + \sigma_b^2 C\right)^{-1} \\
&\qquad\quad (\sigma_M^2(\boldsymbol{x}-\mu_0\mathbf{1}) + \sigma_b^2\tilde{\boldsymbol{d}}) \\
&= \sigma_b^2\rho(x_*-\tilde{x}_*) + \frac{\sigma_b^2|\mathbf{1}^{\mathrm{T}}C^{-1}\mathbf{1}|}{|F^{\mathrm{T}}C^{-1}F|} - \left|\begin{pmatrix} \mathbf{1}^{\mathrm{T}} \\ (\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} \end{pmatrix} \frac{C^{-1}}{\sigma_b^2}(\ \mathbf{1}\quad (\boldsymbol{x}-\mu_0\mathbf{1})\ )\right|^{-1} \\
&\qquad \left(\left|\begin{pmatrix} \mathbf{1}^{\mathrm{T}} \\ (\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} \end{pmatrix} \frac{C^{-1}}{\sigma_b^2}(\ \mathbf{1}\quad \sigma_b^2\tilde{\boldsymbol{d}}\ )\right| + \right. \\
&\qquad \left.\left|\begin{pmatrix} \mathbf{1}^{\mathrm{T}} \\ \sigma_b^2\boldsymbol{d}^{\mathrm{T}} \end{pmatrix} \frac{C^{-1}}{\sigma_b^2}(\ \mathbf{1}\quad (\boldsymbol{x}-\mu_0\mathbf{1})\ )\right| + \right.
\end{aligned}
$$

$$\left|\begin{pmatrix} \mathbf{1}^{\mathrm{T}} \\ (\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} \\ \sigma_b^2\boldsymbol{d}^{\mathrm{T}} \end{pmatrix} \frac{C^{-1}}{\sigma_b^2}\begin{pmatrix} \mathbf{1} & (\boldsymbol{x}-\mu_0\mathbf{1}) & \sigma_b^2\tilde{\boldsymbol{d}} \end{pmatrix}\right|$$

$$= \sigma_b^2\rho(x_*-\tilde{x_*}) + |F^{\mathrm{T}}C^{-1}F|^{-1}\left(\sigma_b^2|\mathbf{1}^{\mathrm{T}}\frac{C^{-1}}{\sigma_b^2}\mathbf{1}| - |F^{\mathrm{T}}\frac{C^{-1}}{\sigma_b^2}\begin{pmatrix} \mathbf{1} & \sigma_b^2\tilde{\boldsymbol{d}} \end{pmatrix}|\right.-$$

$$\left.\left|\begin{pmatrix} \mathbf{1}^{\mathrm{T}} \\ \sigma_b^2\boldsymbol{d}^{\mathrm{T}} \end{pmatrix}\frac{C^{-1}}{\sigma_b^2}F\right| - \frac{1}{\sigma_b^2}\left|\begin{pmatrix} F^{\mathrm{T}} \\ \sigma_b^2\boldsymbol{d}^{\mathrm{T}} \end{pmatrix}\frac{C^{-1}}{\sigma_b^2}\begin{pmatrix} F & \sigma_b^2\tilde{\boldsymbol{d}} \end{pmatrix}\right|\right)$$

$$= \sigma_b^2\rho(x_*-\tilde{x_*}) - \sigma_b^2\frac{\begin{vmatrix} F^{\mathrm{T}}C^{-1}F & F^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{d}} - \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ \boldsymbol{d}^{\mathrm{T}}C^{-1}F - (0\ 1) & \boldsymbol{d}^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{d}} \end{vmatrix}}{|F^{\mathrm{T}}C^{-1}F|}$$

$$= \sigma_b^2\left(\rho(x_*-\tilde{x_*}) - \boldsymbol{d}^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{d}}+\right.$$

$$\left.[\boldsymbol{d}^{\mathrm{T}}C^{-1}F - (0\ 1)](F^{\mathrm{T}}C^{-1}F)^{-1}\left[F^{\mathrm{T}}C^{-1}\tilde{\boldsymbol{d}} - \begin{pmatrix} 0 \\ 1 \end{pmatrix}\right]\right) \tag{B.14}$$

$\mathrm{Cov}([B(x_*)/\boldsymbol{Y}],[Y(\tilde{x_*})/\boldsymbol{Y}])$

$$= \lim_{\sigma_M\to\infty}\sigma_M^2(\tilde{x_*}-\mu_0) + \sigma_b^2(\tilde{x_*}-\mu_0)\rho(x_*-\tilde{x_*}) - (\sigma_M^2(\boldsymbol{x}-\mu_0\mathbf{1}) + \sigma_b\boldsymbol{d})^{\mathrm{T}}$$
$$\left(\sigma_M^2(\boldsymbol{x}-\mu_0\mathbf{1})(\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} + (b^2+\sigma_M^2)\sigma_0^2\mathbf{1}\mathbf{1}^{\mathrm{T}} + \sigma_b^2C\right)^{-1}$$
$$\left(\sigma_M^2(\tilde{x_*}-\mu_0)(\boldsymbol{x}-\mu_0\mathbf{1}) + (b^2+\sigma_M^2)\sigma_0^2\mathbf{1} + \sigma_b^2\tilde{\boldsymbol{c}}\right)$$

$$= \sigma_b^2(\tilde{x_*}-\mu_0)\rho(x_*-\tilde{x_*}) - \left|\begin{pmatrix} \mathbf{1}^{\mathrm{T}} \\ (\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} \end{pmatrix}\frac{C^{-1}}{\sigma_b^2}\begin{pmatrix} \mathbf{1} & (\boldsymbol{x}-\mu_0\mathbf{1}) \end{pmatrix}\right|^{-1}$$
$$\left((x_*-\mu_0)|\mathbf{1}^{\mathrm{T}}\frac{C^{-1}}{\sigma_b^2}\mathbf{1}| - (x_*-\mu_0)\left|\begin{pmatrix} \mathbf{1}^{\mathrm{T}} \\ \sigma_b^2\boldsymbol{d}^{\mathrm{T}} \end{pmatrix}\frac{C^{-1}}{\sigma_b^2}\begin{pmatrix} \mathbf{1} & (\boldsymbol{x}-\mu_0\mathbf{1}) \end{pmatrix}\right|-\right.$$
$$\left.|(\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}}\frac{C^{-1}}{\sigma_b^2}\mathbf{1}| - \left|\begin{pmatrix} (\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} \\ \sigma_b^2\boldsymbol{d}^{\mathrm{T}} \end{pmatrix}\frac{C^{-1}}{\sigma_b^2}\begin{pmatrix} \mathbf{1} & (\boldsymbol{x}-\mu_0\mathbf{1}) \end{pmatrix}\right|-\right.$$
$$\left.\left|\begin{pmatrix} \mathbf{1}^{\mathrm{T}} \\ (\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} \end{pmatrix}\frac{C^{-1}}{\sigma_b^2}\begin{pmatrix} \mathbf{1} & \sigma_b^2\tilde{\boldsymbol{c}} \end{pmatrix}\right|-\right.$$

$$
\left|\begin{pmatrix} \mathbf{1}^{\mathrm{T}} \\ (\boldsymbol{x}-\mu_0\mathbf{1})^{\mathrm{T}} \\ \sigma_b^2\boldsymbol{d}^{\mathrm{T}} \end{pmatrix} \frac{C^{-1}}{\sigma_b^2} \begin{pmatrix} 1 & (\boldsymbol{x}-\mu_0\mathbf{1}) & \sigma_b^2\tilde{\mathbf{c}} \end{pmatrix}\right|\Bigg)
$$

$$
\begin{aligned}
= \; & \sigma_b^2(\tilde{x}_* - \mu_0)\rho(x_* - \tilde{x}_*) - |F^{\mathrm{T}}C^{-1}F|^{-1} \\
& \Big((\tilde{x}_* - \mu_0)|\sigma_b^2\mathbf{1}^{\mathrm{T}}C^{-1}\mathbf{1}| - \sigma_b^2|\boldsymbol{x}^{\mathrm{T}}C^{-1}\mathbf{1}| + \sigma_b^2\mu_0|\mathbf{1}^{\mathrm{T}}C^{-1}\mathbf{1}| - \\
& |F^{\mathrm{T}}C^{-1}\begin{pmatrix} 1 & \sigma_b^2\tilde{\mathbf{c}} \end{pmatrix}| - (\tilde{x}_* - \mu_0)\left|\begin{pmatrix} \mathbf{1}^{\mathrm{T}} \\ \sigma_b^2\boldsymbol{d}^{\mathrm{T}} \end{pmatrix} C^{-1}F\right| - \\
& \left|\begin{pmatrix} \boldsymbol{x}^{\mathrm{T}} \\ \sigma_b^2\boldsymbol{d}^{\mathrm{T}} \end{pmatrix} C^{-1}F\right| + \mu_0\left|\begin{pmatrix} \mathbf{1}^{\mathrm{T}} \\ \sigma_b^2\boldsymbol{d}^{\mathrm{T}} \end{pmatrix} C^{-1}F\right| - \\
& \frac{1}{\sigma_b^2}\left|\begin{pmatrix} F^{\mathrm{T}} \\ \sigma_b^2\boldsymbol{d}^{\mathrm{T}} \end{pmatrix} C^{-1}\begin{pmatrix} F & \sigma_b^2\tilde{\mathbf{c}} \end{pmatrix}\right|\Big)
\end{aligned}
$$

$$
= \sigma_b^2(\tilde{x}_* - \mu_0)\rho(x_* - \tilde{x}_*) - \sigma_b^2\frac{\begin{vmatrix} F^{\mathrm{T}}C^{-1}F & F^{\mathrm{T}}C^{-1}\tilde{\mathbf{c}} - \tilde{\boldsymbol{f}} \\ \boldsymbol{d}^{\mathrm{T}}C^{-1}F - (0\ 1) & \boldsymbol{d}^{\mathrm{T}}C^{-1}\tilde{\mathbf{c}} \end{vmatrix}}{|F^{\mathrm{T}}C^{-1}F|}
$$

$$
\begin{aligned}
= \; & \sigma_b^2\Big((\tilde{x}_* - \mu_0)\rho(x_* - \tilde{x}_*) - \boldsymbol{d}^{\mathrm{T}}C^{-1}\tilde{\mathbf{c}} + \\
& (\boldsymbol{d}^{\mathrm{T}}C^{-1}F - (0\ 1))(F^{\mathrm{T}}C^{-1}F)^{-1}(F^{\mathrm{T}}C^{-1}\tilde{\mathbf{c}} - \tilde{\boldsymbol{f}})\Big)
\end{aligned}
\tag{B.15}
$$

Appendix C

C Sources For Programs

C.1 trace.c

This routine produces the example traces of the functions shown in Figures 5.5, 5.25, 5.26 and 5.27. The assymetry of these graphs comes from rounding errors in this program, which develop assymetrically, as the choleski decompostion starts at the left.

```
#include <math.h>
#include <stdio.h>
#define X_range 2.0
#define No_of_points 11
#define Array_size No_of_points+1
#define Delta_x 2*X_range/(No_of_points-1)
#define No_of_traces 10
#define Pi2 6.28318530717958647692528676655900576839433879875
#define Pi  3.14159265358979323846264338327950288419716939938

FILE *fopen(),*outfile;
long int seed=1003;
double m[Array_size][Array_size],u[Array_size][Array_size];

/*****************************************************************************/
/* Generate a uniform [0,x] random variable                                  */
/*****************************************************************************/

double uniform(double x)
{
  seed=(75+seed*1741)%65537;
  return(x*seed/65537);
}

/*****************************************************************************/
/* Using a Uniform[0,1] and a Uniform[2pi] generate a Normal(0,1) random variable */
/*****************************************************************************/
```

```
double normal()
{
  return(sqrt(-2.0*log(uniform(1.0)))*cos(uniform(Pi2)));
}

/*****************************************************************************/
/* Generate a vector of independent Normal(0,1) random variables             */
/*****************************************************************************/

generate(double vec[Array_size])
{
  int ll;
  for(ll=1;ll<=No_of_points;ll++)
    vec[ll]=normal();
  return;
}

/*****************************************************************************/
/* Map theta to its "area" equivalent to get compable traces                 */
/*****************************************************************************/

double thetaa(double theta,int rho_no)
{
  switch(rho_no)
    {
    case 1 : return(theta);
    case 2 : return(2*sqrt(theta/Pi));
    case 3 : return(sqrt(Pi/theta));
    case 4 : return(4*sqrt(Pi/theta)/3);
    }
}

/*****************************************************************************/
/* Output a vector of Normal(0,U) random variables                           */
/*****************************************************************************/

output(double vec[Array_size],double lam)
{
  double t;
  int ll,l;
  for(ll=1;ll<=No_of_points;ll++)
    {
      t=lam*(-X_range+(ll-1)*Delta_x);
      for(l=ll+1;l<=Array_size;l++)
        t+=(u[ll][l]*vec[l]);
      fprintf(outfile,"%20.10f ",t);
    }
  fprintf(outfile,"\n");
  return;
}

/*****************************************************************************/
/* Choleski decompose A, into upper triangular matrix B                      */
/*****************************************************************************/

choleski(double a[Array_size][Array_size],double b[Array_size][Array_size])
{
  int i,j,k;
  double t;

  b[1][1]=sqrt(a[1][1]);
  for(j=2;j<=No_of_points;j++)
    b[1][j]=a[1][j]/b[1][1];
  for(i=2;i<No_of_points;i++)
    {
```

```
      t=a[i][i];
      for(k=1;k<i;k++)
        t-=(b[k][i]*b[k][i]);
      if(t>0)
        b[i][i]=sqrt(t);
      else
        b[i][i]=0;
      for(j=i+1;j<=No_of_points;j++)
        {
          t=a[i][j];
          for(k=1;k<i;k++)
            t-=(b[k][i]*b[k][j]);
          if(b[i][i]!=0)
            b[i][j]=t/b[i][i];
          else
            b[i][j]=0;
        }
    }
  t=a[No_of_points][No_of_points];
  for(k=1;k<No_of_points;k++)
    t-=(b[k][No_of_points]*b[k][No_of_points]);
  b[No_of_points][No_of_points]=sqrt(t);
  return;
}

/*****************************************************************************/
/* Raise 10 to the power n                                                   */
/*****************************************************************************/

double ten(int n)
{
  return(exp(log(10.0)*n));
}

/*****************************************************************************/
/* Calculate the covariance between Y(x) and Y(y), b=lam, s0=sb=1, mu0=0     */
/* for ''slope'' covariance p2 -- n=1; p1 -- n=2; p1+ -- n=3; pc+ -- n=4.    */
/*****************************************************************************/

double cov(double x,double y,double th,double lam,int n)
{
  switch(n)
    {
    case 1:return( lam*lam+(1+x*y)*exp(-th*(x-y)*(x-y)) );
    case 2:return( lam*lam+(1+x*y)*exp(-th*fabs(x-y)) );
    case 3:
      if(fabs(x-y)<th)
        return( lam*lam+(1+x*y)*(1-fabs(x-y)/th) );
      else
        return( lam*lam);
    case 4:
      if(fabs(x-y)<(th/2))
        return( lam*lam+(1+x*y)*(1-6*(x-y)*(x-y)*(1-fabs(x-y)/th)/th/th) );
      else
        {
          if(fabs(x-y)<th)
            return( lam*lam+(1+x*y)*2*(1-fabs(x-y)/th)*(1-fabs(x-y)/th)*(1-fabs(x-y)/th) );
          else
            return( lam*lam );
        }
    }
}

/*****************************************************************************/
/* Main block, produce tables of traces of Y(x), for all four covariance     */
/* structures and for lam=0.1,1,10, and th=0.1,1,10                          */
```

```
/*****************************************************************************/

main(int argc,char *argv[])
{
  int logth,loglam,nnn;
  int loop1,loop2,loop3;
  double y[Array_size];
  double th,lam;
  double temp;

  if (argc<=1)
    {
      fprintf(stderr,"NOT ENOUGH FILES\n"); exit(2);
    }
  else
    if((outfile=fopen(*++argv,"w"))==NULL)
      {
        fprintf(stderr,"OUTPUT FILE ERROR\n");
        exit(2);
      }
  for(nnn=1;nnn<5;nnn++)
  for(logth=-1;logth<2;logth++)
    for(loglam=-1;loglam<2;loglam++)
      {
        th=ten( logth);lam=ten(loglam);
        for(loop1=1;loop1<=No_of_points;loop1++)
          for(loop2=1;loop2<=No_of_points;loop2++)
            m[loop1][loop2]=cov(-X_range+loop1*Delta_x,-X_range+loop2*Delta_x,thetaa(th,nnn),lam,nnn);
        printf("%2d %2d %2d ",logth,loglam,nnn);
        choleski(m,u);
        for(loop1=1;loop1<=No_of_traces;loop1++)
          {
            generate(y);
            output(y,lam);
            fprintf(outfile,"\n");
          }
        printf("\n");
      }
}
```

C.2 eqn.c

This program, was the original program developed to generate optimal designs, and is a direct translation of the original FORTRAN77 source,

```
#include <stdio.h>
#include <math.h>
#define MAX_SIZE 16
#define PREC 0.00000000001
#define END_CONDITION 0.000001
#define J_ABS(X)              ( (X) > 0   ? (X) : -(X) )
#define J_MAX(X,Y)            ( (X) > (Y) ? (X) :  (Y) )
#define GOLDEN_RATIO    0.618033988749894848204586834365638 54
#define GOLDEN_RATIO_C  0.381966011250105151795413165634361 46
double c[MAX_SIZE][MAX_SIZE],ca[MAX_SIZE][MAX_SIZE],
       cm[MAX_SIZE][MAX_SIZE],y[MAX_SIZE],x[MAX_SIZE],xx[MAX_SIZE],
       c_inv_1[MAX_SIZE],mu,si_0,b,si_b,th,lam,t,si_0_0,si_b_0,mu_star;
```

```
int    cp[MAX_SIZE],n,crit_no,mu_no,var_no,fun_no;
FILE *fopen(),*file_y,*file_crit;
int out_count,count_out,y_flag,crit_flag;

/************************************/
/* Protect against errors produced  */
/* by taking the square root of a   */
/* negative real                    */
/************************************/

double sqroot(val)
double val;
{
  if(val>=0)
    return(sqrt(val));
  else
    {
      printf("SQRT Error %f\n",val);
      return(sqrt(-val));
    }
}

/************************************/
/* Function to be approximated      */
/************************************/

double f(double val)
{

  switch(fun_no)
    {
    case 1 : return(sin(val/2-.1));
    case 2 : return((val-0.2)*(val-0.2)*(val-0.2)*(val-0.2)*(val-0.2)*
    (val-0.2)*(val-0.2)*(val-0.2)*(val-0.2)*(val-0.2)*(val-0.2)+0.01977326743);
    case 3 : return((val<0)?1:((val>2)?-1:1-val));
    case 4 : return((val-.5)*(val-.5)*exp(val));
    case 5 : return((val-.5)*(val-.5)*exp(val)+.1);
    case 6 : return((val-.5)*(val-.5)*exp(val)-.1);
    }
}

/************************************/
/* Predictor of Y at x_2            */
/************************************/

double yyyyy(double x_2)
{
  int l1;
  double p_y_y,c_star_c_inv_1,c_star[MAX_SIZE],c_inv_c_star[MAX_SIZE],tv[MAX_SIZE];

  for(l1=1;l1<=n;l1++)
    c_star[l1]=(1+(x_2-mu)*(x[l1]-mu)/si_0)*
      exp(-th*(x_2-x[l1])*(x_2-x[l1]));
  if(n==1)
    c_inv_c_star[1]=c_star[1]/c[1][1];
  else
    solve_y(n,ca,cm,cp,c_star,c_inv_c_star);
  p_y_y=0;
  c_star_c_inv_1=1;
  for(l1=1;l1<=n;l1++)
    {
      p_y_y+=c_inv_1[l1]*(tv[l1]=y[l1]-b*(x[l1]-mu));
      c_star_c_inv_1-=c_inv_c_star[l1];
    }
  p_y_y=p_y_y*c_star_c_inv_1*lam/(1+lam*t)+b*(x_2-mu);
  for(l1=1;l1<=n;l1++)
```

```
    p_y_y+=c_inv_c_star[l1]*tv[l1];
  return(p_y_y);
}

/************************************/
/* Criterion for choosing new point */
/*   crit_no=0 Var[X_0/Y]/Var(X_0)  */
/*     =1 (Py(Y)^2+Var[Y/Y])/Var(Y) */
/*     =2 Py(Y)^2/Var[Y/Y]          */
/*     =3 (Py(Y)^2+Var[Y/Y])        */
/************************************/

double criterion(double x_2)
{
  int l1;
  double p_y_y,var_y_y,var_y,c_star_c_inv_1,
         c_star[MAX_SIZE],c_inv_c_star[MAX_SIZE],tv[MAX_SIZE];

  for(l1=1;l1<=n;l1++)
    c_star[l1]=(1+(x_2-mu)*(x[l1]-mu)/si_0)*
      exp(-th*(x_2-x[l1])*(x_2-x[l1]));
  if(n==1)
    c_inv_c_star[1]=c_star[1]/c[1][1];
  else
    solve_y(n,ca,cm,cp,c_star,c_inv_c_star);
  if(crit_no==0)
    {
      c_star_c_inv_1=1;
      p_y_y=1+(x_2-mu)*(x_2-mu);
      for(l1=1;l1<=n;l1++)
        {
          p_y_y-=c_inv_c_star[l1]*c_star[l1];
          c_star_c_inv_1-=c_inv_c_star[l1];
        }
     if(p_y_y<0.00000000000000000000000000001)
         return(0);
     else
         return(c_star_c_inv_1*c_star_c_inv_1/p_y_y);
    }
  else
    {
      p_y_y=0;
      c_star_c_inv_1=1;
      for(l1=1;l1<=n;l1++)
        {
          p_y_y+=c_inv_1[l1]*(tv[l1]=y[l1]-b*(x[l1]-mu));
          c_star_c_inv_1-=c_inv_c_star[l1];
        }
      p_y_y=p_y_y*c_star_c_inv_1*lam/(1+lam*t)+b*(x_2-mu);
      for(l1=1;l1<=n;l1++)
          p_y_y+=c_inv_c_star[l1]*tv[l1];
      var_y_y=(var_y=si_0*si_b*(1+(x_2-mu)*(x_2-mu)/si_0+lam))-si_0*si_b*
        (lam+c_star_c_inv_1*c_star_c_inv_1/(1+lam*t));
      for(l1=1;l1<=n;l1++)
        var_y_y-=si_0*si_b*c_star[l1]*c_inv_c_star[l1];
      if(var_y_y<0.00000000000000000000000000000001)
        var_y_y=0.00000000000000000000000000000001;
      switch(crit_no)
        {
        case 1 : return(-(p_y_y*p_y_y+var_y_y)/var_y);
        case 2 : return(-p_y_y*p_y_y);
        case 3 : return(-(p_y_y*p_y_y+var_y_y));
        }
    }
}
```

```
/************************************/
/* Finds the maximum value of       */
/* criterion from the set           */
/* {xs+prec,xs+2prec,...,xf-prec}   */
/************************************/

double golden(double x_start,double x_finish,double precision)
{
  double max_x,max_y,ptr_y,ptr_x;
  int l1,total;
  double x1,x2,x3,x4,y1,y2,y3,y4;

  if(fabs(x_start-x_finish)<=precision)
    return(-999999);
  x1=x_start;
  x4=x_finish;
  x3=GOLDEN_RATIO*x_finish+GOLDEN_RATIO_C*x_start;
  x2=GOLDEN_RATIO_C*x_finish+GOLDEN_RATIO*x_start;
  y2=criterion(x2);
  y3=criterion(x3);
  y1=criterion(x1);
  y4=criterion(x4);
  do
    {
      if( J_MAX(y1,y2)>J_MAX(y3,y4) )
        {x4=x3;y4=y3;x3=x2;y3=y2;
         x2=GOLDEN_RATIO_C*x4+GOLDEN_RATIO*x1;y2=criterion(x2);}
      else
        {x1=x2;y1=y2;x2=x3;y2=y3;
         x3=GOLDEN_RATIO*x4+GOLDEN_RATIO_C*x1;y3=criterion(x3);}
    }
  while((x4-x1)>precision);
  if(y1>=y2 && y1>=y3 && y1>=y4)
    return(x1);
  if(y2>=y1&& y2>=y3 && y2>=y4)
    return(x2);
  if(y3>=y2 && y3>=y1 && y3>=y4)
    return(x3);
  if(y4>=y2 && y4>=y3 && y4>=y1)
    return(x4);
}

/************************************/
/* Finds the maximum value of       */
/* criterion from the set           */
/* {sx,sx+prec,...,xf-prec,xf}\     */
/*  {x_1,...,x_n}                   */
/************************************/

double maximize_x(double start_x,double finish_x,double precision)
{
  double x_max,y_max,x_ptr,y_ptr;
  int l1;

  x_max=golden(start_x,xx[1],precision);
  if(x_max<-10)
    y_max=-99999999;
  else
    y_max=criterion(x_max);
  if(n>1)
    for(l1=1;l1<=n-1;l1++)
      {
        x_ptr=golden(xx[l1],xx[l1+1],precision);
        if(x_ptr<-10)
          y_ptr=-99999999;
        else
```

```
          y_ptr=criterion(x_ptr);
        if(y_ptr>y_max)
          {
            y_max=y_ptr;
            x_max=x_ptr;
          }
      }
  x_ptr=golden(xx[n],finish_x,precision);
  if(x_ptr<-10)
    y_ptr=-99999999;
  else
    y_ptr=criterion(x_ptr);
  if(y_ptr>y_max)
    x_max=x_ptr;
  return(x_max);
}

/************************************/
/* Read in the parameter values and */
/* initialize variables             */
/************************************/

setup()
{
  n=1;
  scanf("%d %d %d %d %le %le %le %le %le",
        &crit_no,&mu_no,&var_no,&fun_no,
        &mu,&si_0,&b,&si_b,&th);
  lam=b*b/si_b;
  si_b_0=si_b;
  si_0_0=si_0;
  c[1][1]=c_inv_1[1]=t=1;
  y[1]=f(xx[1]=x[1]=mu);
  return;
}

/************************************/
/* Compute Py(X_0)                  */
/* update mu_0                      */
/* mu_no=1 put mu_0 = Py(X_0)       */
/*    =2 put mu_0 = arg min Crit(x) */
/*    =something else = do nothing  */
/* update b                         */
/* mu_no=0 do nothing               */
/*    !=0 put b = Py(B(mu_0))       */
/* update Si_0 and Si_b             */
/* var_no=0 do nothing              */
/*    =1 put Si_0 = Var[X_0/Y]      */
/*           Si_b = Var[b/Y]        */
/*    =2 put Si_0 and Si_b in same  */
/*   ratio but so that Si_0Si_b = s */
/************************************/

int compute_mean()
{
  double temp,tv1[MAX_SIZE],tv2[MAX_SIZE],ratio_0,ratio_b,
         d[MAX_SIZE],c_inv_d[MAX_SIZE],si_0_si_b;
  int l1,l2;

  temp=mu_star=mu;
  for(l1=1;l1<=n;l1++)
    mu_star-=c_inv_1[l1]*lam*(y[l1]/b-x[l1]);
  mu_star/=(1+lam*t);
  switch(mu_no)
    {
    case 2 : {mu=maximize_x(-3.0,3.0,PREC);break;}
```

```
    case 1 : {mu=mu_star;break;}
    default : {printf("%10.5f \n",mu_star);break;}
    }
  if(mu_no!=0)
    {
      for(l1=1;l1<=n;l1++)
        tv1[l1]=y[l1]-b*(x[l1]-mu_star);
      if(n==1)
        tv2[1]=tv1[1];
      else
        solve_y(n,ca,cm,cp,tv1,tv2);
      for(l1=1;l1<=n;l1++)
        b+=(x[l1]-temp)*exp(-th*(x[l1]-mu)*(x[l1]-mu))*tv2[l1]/si_0;
      lam=b*b/si_b;
      for(l1=1;l1<=n;l1++)
        for(tv1[l1]=l2=1;l2<=n;l2++)
          c[l1][l2]=(1+(x[l1]-mu)*(x[l2]-mu)/si_0)
            *exp(-th*(x[l1]-x[l2])*(x[l1]-x[l2]));
      if(solve_0(c,n,ca,cm,cp))
        return(1);
      else
        {
          solve_y(n,ca,cm,cp,tv1,c_inv_1);
          t=0;
          for(l1=1;l1<=n;l1++)
            t+=c_inv_1[l1];
        }
    }
  if(var_no>0)
    {
      ratio_0=1/(1+lam*t);
      ratio_b=1;
      temp=0;
      for(l1=1;l1<=n;l1++)
        d[l1]=(x[l1]-mu)*exp(-th*(x[l1]-mu)*(x[l1]-mu))/sqroot(si_0);
      solve_y(n,ca,cm,cp,d,c_inv_d);
      for(l1=1;l1<=n;l1++)
        {
          ratio_b-=d[l1]*c_inv_d[l1];
          temp+=c_inv_d[l1];
        }
      ratio_b+=lam*temp*temp*ratio_0;
      if(var_no==1)
        {
          si_b=si_b_0*ratio_b;
          si_0=si_0_0*ratio_0;
        }
      else
        {
          if(n<2)
            si_0_si_b=si_0*si_b;
          else
            {
              si_0_si_b=temp=0;
              for(l1=1;l1<=n;l1++)
                temp+=x[l1]-y[l1]/b;
              temp/=n;
              for(l1=1;l1<=n;l1++)
                tv1[l1]=(x[l1]-y[l1]/b-temp);
              solve_y(n,ca,cm,cp,tv1,tv2);
              for(l1=1;l1<=n;l1++)
                si_0_si_b+=tv1[l1]*c_inv_1[l1];
              si_0_si_b*=-si_0_si_b/t;
              for(l1=1;l1<=n;l1++)
                si_0_si_b+=tv2[l1]*tv1[l1];
              si_0_si_b/=(b*b*(n-1));
```

```
          }
        si_b=sqroot(si_0_si_b*ratio_b/ratio_0*si_b_0/si_0_0);
        if(si_b==0)
            si_0_si_b=(si_0=si_0_0)*(si_b=si_b_0);
        si_0=J_ABS(si_0_si_b/si_b);
      }
    lam=b*b/si_b;
    for(l1=1;l1<=n;l1++)
      for(tv1[l1]=l2=1;l2<=n;l2++)
        c[l1][l2]=(1+(x[l1]-mu)*(x[l2]-mu)/si_0)
          *exp(-th*(x[l1]-x[l2])*(x[l1]-x[l2]));
    if(solve_0(c,n,ca,cm,cp))
      return(1);
    else
      {
        solve_y(n,ca,cm,cp,tv1,c_inv_1);
        t=0;
        for(l1=1;l1<=n;l1++)
          t+=c_inv_1[l1];
      }
  }
 return(0);
}

/************************************/
/* Add x_2 to the design and        */
/* evaluate f(x_2)                  */
/************************************/

int add_x(double x_2)
{
 int ptr,l1,l2,error_flag;
 double tv[MAX_SIZE];

 if(x_2>xx[n])
   xx[n+1]=x_2;
 else
   {
     ptr=n;
     for(ptr=n;(ptr>0 && x_2<xx[ptr]);ptr--)
         xx[ptr+1]=xx[ptr];
     xx[ptr+1]=x_2;
   }
 n++;
 for(l1=1;l1<=n-1;l1++)
   c[n][l1]=c[l1][n]=(1+(x_2-mu)*(x[l1]-mu)/si_0)
     *exp(-th*(x_2-x[l1])*(x_2-x[l1]));
 c[n][n]=1+(x_2-mu)*(x_2-mu)/si_0;
 x[n]=x_2;
 y[n]=f(x_2);
 for(l1=1;l1<=n;l1++)
   tv[l1]=1;
 if(!(error_flag=solve_0(c,n,ca,cm,cp)))
   {
     solve_y(n,ca,cm,cp,tv,c_inv_1);
     t=0;
     for(l1=1;l1<=n;l1++)
       t+=c_inv_1[l1];
   }
 return(error_flag);
}

/************************************/
/* Matrix inversion routines        */
/* preliminary routine -- solve_0   */
/* solve for vector    -- solve_y   */
```

```
/************************************/

int solve_O(mat,mat_size,mat_a,mat_m,mat_p)
     double mat[MAX_SIZE][MAX_SIZE], mat_a[MAX_SIZE][MAX_SIZE], mat_m[MAX_SIZE][MAX_SIZE];
     int mat_size, mat_p[MAX_SIZE];

{
  double s[MAX_SIZE],d_ptr,d_star;
  int l1,l2,l3,ptr,temp_i,rows[MAX_SIZE],error_flag=0;

  for(l1=1; l1<=mat_size; l1++)
    {
      s[(rows[l1]=l1)]=fabs(mat_a[l1][1]=mat[l1][1]);
      for(l2=2; l2<=mat_size; l2++)
        if(fabs(mat_a[l1][l2]=mat[l1][l2])>s[l1])
            s[l1]=J_ABS(mat[l1][l2]);
      if(s[l1]==0)
        {
          error_flag=1;
          break;
        }
      }
  if(error_flag)
    return(error_flag);
  for(l1=1; l1<=mat_size-1; l1++)
    {
      d_ptr=fabs(mat_a[rows[(ptr=l1)]][l1])/s[rows[l1]];
      for(l2=l1+1; l2<=mat_size; l2++)
        if((d_star=fabs(mat_a[rows[l2]][l1])/s[rows[l2]])>d_ptr)
          {
            d_ptr=d_star;
            ptr=l2;
          }
     if(mat_a[rows[ptr]][l1]==0)
        {
          error_flag=1;
          break;
        }
     if((mat_p[l1]=ptr)!=l1)
       {
         temp_i=rows[l1];
         rows[l1]=rows[ptr];
         rows[ptr]=temp_i;
       }
     for(l2=l1+1; l2<=mat_size; l2++)
       {
         mat_m[rows[l2]][l1]=mat_a[rows[l2]][l1]/mat_a[rows[l1]][l1];
         for(l3=l1+1; l3<=mat_size; l3++)
           mat_a[rows[l2]][l3]-=(mat_m[rows[l2]][l1]*mat_a[rows[l1]][l3]);
       }
    }
  if(error_flag)
    return(error_flag);
  return(mat_a[rows[mat_size]][mat_size]==0);
}

solve_y(mat_size,mat_a,mat_m,mat_p,rhs,solution)
     double mat_a[][MAX_SIZE], mat_m[][MAX_SIZE], rhs[], solution[];
     int mat_size, mat_p[];

{
  double rhs_a[MAX_SIZE];
  int temp_i,l1,l2,ptr,rows[MAX_SIZE];

  for(l1=1; l1<=mat_size; l1++)
    rhs_a[(rows[l1]=l1)]=rhs[l1];
```

```
  for(l1=1; l1<=mat_size-1; l1++)
    {
      if((ptr=mat_p[l1])!=l1)
        {
          temp_i=rows[l1];
          rows[l1]=rows[ptr];
          rows[ptr]=temp_i;
        }
      for(l2=l1+1; l2<=mat_size; l2++)
        rhs_a[rows[l2]]-=mat_m[rows[l2]][l1]*rhs_a[rows[l1]];
    }
  solution[mat_size]=rhs_a[rows[mat_size]]/mat_a[rows[mat_size]][mat_size];
  for(l1=mat_size-1; l1>=1; l1--)
    {
      solution[l1]=rhs_a[rows[l1]];
      for(l2=l1+1; l2<=mat_size; l2++)
        solution[l1]-=mat_a[rows[l1]][l2]*solution[l2];
      solution[l1]/=mat_a[rows[l1]][l1];
    }
  return;
}

/************************************/
/* print value to FILE *fp, in rows */
/* of five, for readability         */
/************************************/

file_print_f(FILE *fp,double value)

{
  fprintf(fp,"%15.7f ",value);
  if((++out_count)==5)
    {
      fprintf(fp,"\n");
      out_count=0;
    }
}

/************************************/
/* header routine to link altogether*/
/************************************/
main(argc,argv)
     int argc;
     char *argv[];
{
  int loop,finish,error;
  double x_loop;
  char runf;
  if (argc == 1)
    y_flag=crit_flag=0;
  else
    {
      switch((*++argv)[0])
        {
        case 'y' : {y_flag=1;crit_flag=0;break;}
        case 'c' : {y_flag=0;crit_flag=1;break;}
        case 'b' : {y_flag=crit_flag=1;break;}
        default  : {y_flag=crit_flag=0;break;}
        }
    }
  --argc;
  if (y_flag==1)
    {
      if(!(--argc) || (file_y = fopen(*++argv,"w")) == NULL)
        {
          fprintf(stderr,"OUTPUT FILE CREATION ERROR: %d\n",argc);
```

```
        exit(2);
      }
  }
if (crit_flag==1)
  {
    if(!(--argc) || (file_crit = fopen(*++argv,"w")) == NULL)
      {
        fprintf(stderr,"OUTPUT FILE CREATION ERROR : %d\n",argc);
        exit(2);
      }
  }
do
  {
    setup();
    if(crit_no>=0)
      {
        printf("Criterion_no: %d Mean_adj_no: %d Var_adj_no: %d Function_no: %d\n",
                crit_no,         mu_no,          var_no,        fun_no);
        printf("Mu : %6.3f Si_0: %6.3f b: %6.3f Si_b: %6.3f Theta: %6.3f\n",mu,si_0,b,si_b,th);
        printf("        X_n          Y_n          Mu_0          b           Si_0          Si_b\n");
        if(y_flag)
          {
            count_out=out_count=0;
            for(x_loop=-3;x_loop<=3.001;x_loop+=.02)
              file_print_f(file_y,x_loop);
            fprintf(file_y,"\n");
            count_out=out_count=0;
            for(x_loop=-3;x_loop<=3.001;x_loop+=.02)
              file_print_f(file_y,f(x_loop));
            fprintf(file_y,"\n");
          }
        if(crit_flag)
          {
            count_out=out_count=0;
            for(x_loop=-3;x_loop<=3.001;x_loop+=.02)
              file_print_f(file_crit,x_loop);
            fprintf(file_crit,"\n");
          }
        for(loop=1;loop<=MAX_SIZE-2;loop++)
          {
            if(error=compute_mean())
              break;
            else
              {
                printf("%2d %12.7f %12.7f %12.7f %12.7f %12.7f %12.7f\n",
                        n,x[n],y[n],mu,b,si_0,si_b);
                if(y_flag)
                  {
                    count_out=out_count=0;
                    for(x_loop=-3;x_loop<=3.001;x_loop+=.02)
                      file_print_f(file_y,yyyyy(x_loop));
                    fprintf(file_y,"\n");
                  }
                if(crit_flag)
                  {
                    count_out=out_count=0;
                    for(x_loop=-3;x_loop<=3.001;x_loop+=.02)
                      file_print_f(file_crit,criterion(x_loop));
                    fprintf(file_crit,"\n");
                  }
                if(finish=(J_ABS(y[n])<END_CONDITION))
                  break;
                else
                  {
                    if(crit_no==6)
                      error=99*add_x(mu_star);
```

```
                    else
                      {
                        if(mu_no==2)
                          error=2*add_x(mu);
                        else
                          error=3*add_x(maximize_x(-3.0,3.0,PREC));
                      }
                  }
                if(error)
                  break;
              }
          }
        if(error)
          printf("Stopped due to singularity error\n");
        else
          {
            if(!finish)
              {
                if(error=4*compute_mean())
                  break;
                else
                  printf("%2d %12.7f %12.7f %12.7f %12.7f %12.7f %12.7f \n",
                         n,x[n],y[n],mu,b,si_0,si_b);
                if(y_flag)
                  {
                    count_out=out_count=0;
                    for(x_loop=-3;x_loop<=3.001;x_loop+=.02)
                      file_print_f(file_y,yyyyy(x_loop));
                    fprintf(file_y,"\n");
                  }
                if(crit_flag)
                  {
                    count_out=out_count=0;
                    for(x_loop=-3;x_loop<=3.001;x_loop+=.02)
                      file_print_f(file_crit,criterion(x_loop));
                    fprintf(file_crit,"\n");
                  }
              }
          }
      }
    if(error)
      {
        printf("Error code : %d\n",error);
        break;
      }
  }
while(crit_no>=0);
}
```

C.3 Newer C sources

The C source is now split into a series of individual blocks which perform various different tasks

C.3.1 prob.h

General include file for new code

```
#include <stdio.h>
#include <math.h>
#define MAX_SIZE 16
#define SIZE_MAX 40
#define PREC 0.00000000001
#define END_CONDITION 0.000001
#define J_ABS(X)              ( (X) > 0   ? (X) : -(X) )
#define J_MAX(X,Y)            ( (X) > (Y) ? (X) :  (Y) )
#define GOLDEN_RATIO    0.61803398874989484820458683436563854
#define GOLDEN_RATIO_C  0.38196601125010515179541316563436146
#define PI       3.14159265358979323846264338327950288
#define SQRT2PI 0.39894228040143267793994605993438163
typedef double MATRIX[MAX_SIZE][MAX_SIZE];
typedef double VECd[MAX_SIZE];
typedef double VECD[SIZE_MAX];
typedef double MATD[SIZE_MAX][SIZE_MAX];
typedef int VECi[MAX_SIZE];
typedef struct matrix
{
  MATRIX m;
  int rows;
  int columns;
} MAT;
typedef struct inv
{
  MATRIX a;
  MATRIX d;
  VECi p;
}
INV;
typedef struct mat_inv
{
  MAT m;
  INV i;
}
MAT_INV;

double rho(double,double);
void setup(int*,int*,int*,int*);
int add_x(double,int);
void file_print(FILE*,double,int*);
void file_print_x(FILE*);
void file_print_f(FILE*,int);
void file_print_crit(FILE*,int);
void file_print_y(FILE*);
double f(double,int);
double P_Y(double);
double Cov_XO_Y(double);
double Cov_B_B(double,double);
double Cov_B_Y(double,double);
double Cov_Y_Y(double,double);
void PY_XO(double*,double*);
void PY_B(double,double,double*,double*);
void PY_Y(double,double,double*,double*);
void PY(double,double,double,
        double*,double*,
        double*,double*,
        double*,double*,double*,
        double*,double*,double*,
```

```
        double*,double*,double*,double*,
        double*,double*,double*);
void PYV(VECD,int,double,VECD,VECD,MATD,MATD,MATD,VECD,VECD,MATD);
void display(MAT*);
int solve_0(MAT_INV*);
void solve_y(MAT_INV*,VECd,VECd);
void solve_m(MAT_INV*,MAT*,MAT*);
double criterion(double,int);
double golden(double,double,double,int);
double maximize_x(double,double,double,int);
void compute_mean();

int n;
double th,mu,s0,b,sb;
MAT_INV v;
VECd x,xx,y;
VECd c1;
double t,mumu,mu_t,s0s0_t;

FILE *fopen(),*file_y,*file_crit;
```

C.3.2 mat2.c

Routines to display a matrix, `display(M)`; invert a square matrix, `solve_0(M)`; solve the set of linear equation $M\boldsymbol{x} = \boldsymbol{y}$, `solve_y(M,y,x)`; and solve the set of linear equation $MA = B$, `solve_m(M,B,A)`. The matrix and vector types are defined in `prob.h`,

```
#include "prob.h"

void display(MAT *m)
{
  int l1,l2;

  for(l1=1;l1<=m->rows;l1++)
    {
      for(l2=1;l2<=m->columns;l2++)
        printf(" %14.2f",m->m[l1][l2]);
      printf("\n");
    }
}

int solve_0(MAT_INV *m)
{
  double s[MAX_SIZE],d_ptr,d_star;
  int l1,l2,l3,ptr,temp_i,rows[MAX_SIZE],error_flag=0;

  if(m->m.rows==1) return(!m->m.m[1][1]);
  for(l1=1; l1<=m->m.rows; l1++)
    {
      s[(rows[l1]=l1)]=fabs(m->i.a[l1][1]=m->m.m[l1][1]);
      for(l2=2; l2<=m->m.rows; l2++)
        if(fabs(m->i.a[l1][l2]=m->m.m[l1][l2])>s[l1])
            s[l1]=fabs(m->m.m[l1][l2]);
      if(s[l1]==0)
        {
```

```
          error_flag=1;
          break;
        }
      }
  if(error_flag)
    {
      display(&(m->m));
      return(error_flag);
    }
  for(l1=1; l1<=m->m.rows-1; l1++)
    {
      d_ptr=fabs(m->i.a[rows[(ptr=l1)]][l1])/s[rows[l1]];
      for(l2=l1+1; l2<=m->m.rows; l2++)
        if((d_star=fabs(m->i.a[rows[l2]][l1])/s[rows[l2]])>d_ptr)
          {
            d_ptr=d_star;
            ptr=l2;
          }
     if(m->i.a[rows[ptr]][l1]==0)
        {
          error_flag=1;
          break;
        }
     if((m->i.p[l1]=ptr)!=l1)
       {
         temp_i=rows[l1];
         rows[l1]=rows[ptr];
         rows[ptr]=temp_i;
       }
     for(l2=l1+1; l2<=m->m.rows; l2++)
       {
         m->i.d[rows[l2]][l1]=m->i.a[rows[l2]][l1]/m->i.a[rows[l1]][l1];
         for(l3=l1+1; l3<=m->m.rows; l3++)
           m->i.a[rows[l2]][l3]-=(m->i.d[rows[l2]][l1]*m->i.a[rows[l1]][l3]);
       }
    }
  if(error_flag)
    {
      display(&(m->m));
      return(error_flag);
    }
  error_flag=(m->i.a[rows[m->m.rows]][m->m.rows]==0);
  if(error_flag)
    display(&(m->m));
  return(error_flag);
}

void solve_y(MAT_INV *m,VECd rhs,VECd solution)
{
  VECd rhs_a;
  int temp_i,l1,l2,ptr,rows[MAX_SIZE];

  if(m->m.rows==1)
    solution[1]=rhs[1]/m->m.m[1][1];
  else
    {
      for(l1=1; l1<=m->m.rows; l1++)
        rhs_a[(rows[l1]=l1)]=rhs[l1];
      for(l1=1; l1<=m->m.rows-1; l1++)
        {
          if((ptr=m->i.p[l1])!=l1)
            {
              temp_i=rows[l1];
              rows[l1]=rows[ptr];
              rows[ptr]=temp_i;
            }
```

```
          for(l2=l1+1; l2<=m->m.rows; l2++)
            rhs_a[rows[l2]]-=m->i.d[rows[l2]][l1]*rhs_a[rows[l1]];
        }
      solution[m->m.rows]=rhs_a[rows[m->m.rows]]/m->i.a[rows[m->m.rows]][m->m.rows];
      for(l1=m->m.rows-1; l1>=1; l1--)
        {
          solution[l1]=rhs_a[rows[l1]];
          for(l2=l1+1; l2<=m->m.rows; l2++)
            solution[l1]-=m->i.a[rows[l1]][l2]*solution[l2];
          solution[l1]/=m->i.a[rows[l1]][l1];
        }
    }
  return;
}

void solve_m(MAT_INV *m, MAT *rhs, MAT *solution)
{
  int temp_i,l1,l2,l3,ptr,rows[MAX_SIZE];
  double rhs_m_a[MAX_SIZE][MAX_SIZE];
  if(m->m.rows==1)
    for(l1=1;l1<=rhs->columns;l1++)
      solution->m[1][l1]=rhs->m[1][l1]/m->m.m[1][1];
  else
    {
      for(l1=1; l1<=m->m.rows; l1++)
        {
          rows[l1]=l1;
          for(l2=1; l2<=rhs->columns; l2++)
            rhs_m_a[l1][l2]=rhs->m[l1][l2];
        }
      for(l1=1; l1<=m->m.rows-1; l1++)
        {
          if((ptr=m->i.p[l1])!=l1)
            {
              temp_i=rows[l1];
              rows[l1]=rows[ptr];
              rows[ptr]=temp_i;
            }
          for(l2=l1+1; l2<=m->m.rows; l2++)
            for(l3=1; l3<=rhs->columns; l3++)
              rhs_m_a[rows[l2]][l3]-=(m->i.d[rows[l2]][l1]*rhs_m_a[rows[l1]][l3]);
        }
      for(l3=1; l3<=rhs->columns; l3++)
        solution->m[m->m.rows][l3]=rhs_m_a[rows[m->m.rows]][l3]
          /m->i.a[rows[m->m.rows]][m->m.rows];
      for(l3=1; l3<=rhs->columns; l3++)
        for(l1=m->m.rows-1; l1>=1; l1--)
          {
            solution->m[l1][l3]=rhs_m_a[rows[l1]][l3];
            for(l2=l1+1; l2<=m->m.rows; l2++)
              solution->m[l1][l3]-=(m->i.a[rows[l1]][l2]*solution->m[l2][l3]);
            solution->m[l1][l3]/=m->i.a[rows[l1]][l1];
          }
    }
  solution->rows=rhs->rows;
  solution->columns=rhs->columns;
  return;
}
```

C.3.3 rho1.c, rho2.c, rho3.c, rho4.c and cov1.c

C sources for the "slope" correlation function $\rho(\cdot)$ - `rho1.c`,...,`rho1.c`; and the process covarince - `cov1.c`.

`rho1.c` - $\rho_2(x)$

```
#include "prob.h"

double rho(double xa,double xb)
{
  return(exp(-th*(xa-xb)*(xa-xb)));
}
```

`rho2.c` - $\rho_1(x)$

```
#include "prob.h"

double rho(double xa,double xb)
{
  double xabs;
  xabs=(xa>xb?(xa-xb):(xb-xa))
  return(exp(-th*xabs));
}
```

`rho3.c` - $\rho_{l+}(x)$

```
#include "prob.h"

double rho(double xa,double xb)
{
  double xabs;
  xabs=(xa>xb?(xa-xb):(xb-xa))/th;
  if(xabs>1) return(0)
    else return(1-xabs);
}
```

`rho4.c` - $\rho_{c+}(x)$

```
#include "prob.h"

double rho(double xa,double xb)
{
  double xabs;
  xabs=(xa>xb?(xa-xb):(xb-xa))/th;
```

```
  if(xabs>1) return(0)
  else if(xabs<.5) return(1-6*xabs*xabs*(1-xabs));
  else return(2*(1-xabs)*(1-xabs)*(1-xabs));
}
```

cov1.c

```
#include "prob.h"

double P_Y(double xxx)
{
  return(b*(xxx-mu));
}

double Cov_XO_Y(double xxx)
{
  return(-b*s0*s0);
}

double Cov_B_B(double xxx,double xxs)
{
  return(rho(xxx,xxs)*sb*sb);
}

double Cov_B_Y(double xxx,double xxs)
{
  return((xxx-mu)*rho(xxx,xxs)*sb*sb);
}

double Cov_Y_Y(double xxx,double xxs)
{
  return((s0*s0+(xxx-mu)*(xxs-mu))*rho(xxx,xxs)*sb*sb+s0*s0*b*b);
}
```

C.3.4 golden.c

Routines to find the maxima of a function which is unimodal golden; or multimodal, but where we know points between which it is unimodal maximize_x.

```
#include "prob.h"

double golden(double x_start,double x_finish,double precision,int crit_no)
{
  double max_x,max_y,ptr_y,ptr_x;
  int l1,total;
  double x1,x2,x3,x4,y1,y2,y3,y4;

  if(fabs(x_start-x_finish)<=precision)
    return(-999999);
  x1=x_start;
  x4=x_finish;
  x3=GOLDEN_RATIO*x_finish+GOLDEN_RATIO_C*x_start;
  x2=GOLDEN_RATIO_C*x_finish+GOLDEN_RATIO*x_start;
  y2=criterion(x2,crit_no);
```

```
  y3=criterion(x3,crit_no);
  y1=criterion(x1,crit_no);
  y4=criterion(x4,crit_no);
  do
    {
      if( J_MAX(y1,y2)>J_MAX(y3,y4) )
        {x4=x3;y4=y3;x3=x2;y3=y2;
         x2=GOLDEN_RATIO_C*x4+GOLDEN_RATIO*x1;y2=criterion(x2,crit_no);}
      else
        {x1=x2;y1=y2;x2=x3;y2=y3;
         x3=GOLDEN_RATIO*x4+GOLDEN_RATIO_C*x1;y3=criterion(x3,crit_no);}
    }
  while((x4-x1)>precision);
  if(y1>=y2 && y1>=y3 && y1>=y4)
    return(x1);
  if(y2>=y1&& y2>=y3 && y2>=y4)
    return(x2);
  if(y3>=y2 && y3>=y1 && y3>=y4)
    return(x3);
  if(y4>=y2 && y4>=y3 && y4>=y1)
    return(x4);
}

/************************************/
/* Finds the maximum value of       */
/* criterion from the set           */
/* {sx,sx+prec,...,xf-prec,xf}\     */
/*  {x_1,...,x_n}                   */
/************************************/

double maximize_x(double start_x,double finish_x,double precision,int crit_no)
{
  double x_max,y_max,x_ptr,y_ptr;
  int l1;

  x_max=golden(start_x,xx[1],precision,crit_no);
  if(x_max<-10)
    y_max=-99999999;
  else
    y_max=criterion(x_max,crit_no);
  if(n>1)
    for(l1=1;l1<=n-1;l1++)
      {
        x_ptr=golden(xx[l1],xx[l1+1],precision,crit_no);
        if(x_ptr<-10)
          y_ptr=-99999999;
        else
          y_ptr=criterion(x_ptr,crit_no);
        if(y_ptr>y_max)
          {
            y_max=y_ptr;
            x_max=x_ptr;
          }
      }
  x_ptr=golden(xx[n],finish_x,precision,crit_no);
  if(x_ptr<-10)
    y_ptr=-99999999;
  else
    y_ptr=criterion(x_ptr,crit_no);
  if(y_ptr>y_max)
    x_max=x_ptr;
  return(x_max);
}
```

C.3.5 update1.c, update2.c, update3.c and update4.c

Routines to compute the new estimates of X_0, $B(\cdot)$, $Y(\cdot)$, and there variances.

update1.c – Extended model

```
#include "prob.h"

void PY_X0(double *mean, double *var)
{
  *mean=mu;
  *var=s0*s0;
  return;
}

double det3(double a11, double a21, double a31,
            double a12, double a22, double a32,
            double a13, double a23, double a33)
{
  return(a11*a22*a33+a21*a32*a13+a31*a12*a23-
         a11*a32*a23-a21*a12*a33-a31*a22*a13);
}

void PY_B(double xxx, double xxs, double *mean, double *cov)
{
  VECd v1,v2,v3,v4;
  int i;
  double det2;
  double m11,m12,m22,m31,m32,p13,p23,p33,v13,v23,v33;

  for(i=1;i<=n;i++)
    {
      v1[i]=1;
      v3[i]=x[i];
    }
  solve_y(&v,v1,v2);
  solve_y(&v,v3,v4);
  m11=m12=m22=p13=p23=0;
  for(i=1;i<=n;i++)
    {
      m11+=v2[i];
      m12+=v4[i];
      m22+=(v4[i]*v3[i]);
      p13+=(v2[i]*y[i]);
      p23+=(v4[i]*y[i]);
    }
  det2=m11*m22-m12*m12;
  for(i=1;i<=n;i++)
    {
      v1[i]=(xxx-mu)*rho(xxx,xxs);
      v3[i]=(xxs-mu)*rho(xxx,xxs);
    }
  solve_y(&v,v1,v2);
  solve_y(&v,v3,v4);
  m31=v13=p33=v33=0;m32=v23=-1;
  for(i=1;i<=n;i++)
    {
      m31+=v2[i];
      v13+=v4[i];
      m32+=(v2[i]*x[i]);
      v23+=(v4[i]*x[i]);
```

```
      p33+=(v2[i]*y[i]);
      v33+=(v2[i]*v3[i]);
    }
  (*cov)=sb*sb*rho(xxx,xxs)-det3(m11,m12,v13,m12,m22,v23,m31,m32,v33)/det2;
  (*mean)=det3(m11,m12,p13,m12,m22,p23,m31,m32,p33)/det2;
  return;
}

void PY_Y(double xxx, double xxs, double *mean, double *cov)
{
   VECd v1,v2,v3,v4;
  int i;
  double det2;
  double m11,m12,m22,m31,m32,p13,p23,p33,v13,v23,v33;

  for(i=1;i<=n;i++)
    {
      v1[i]=1;
      v3[i]=x[i];
    }
  solve_y(&v,v1,v2);
  solve_y(&v,v3,v4);
  m11=m12=m22=p13=p23=0;
  for(i=1;i<=n;i++)
    {
      m11+=v2[i];
      m12+=v4[i];
      m22+=(v4[i]*v3[i]);
      p13+=(v2[i]*y[i]);
      p23+=(v4[i]*y[i]);
    }
  det2=m11*m22-m12*m12;
  for(i=1;i<=n;i++)
    {
      v1[i]=(s0*s0+(xxx-mu)*(x[i]-mu))*rho(xxx,x[i]);
      v3[i]=(s0*s0+(xxs-mu)*(x[i]-mu))*rho(xxs,x[i]);
    }
  solve_y(&v,v1,v2);
  solve_y(&v,v3,v4);
  m31=v13=-1;p33=v33=0;m32=-xxx;v23=-xxs;
  for(i=1;i<=n;i++)
    {
      m31+=v2[i];
      v13+=v4[i];
      m32+=(v2[i]*x[i]);
      v23+=(v4[i]*x[i]);
      p33+=(v2[i]*y[i]);
      v33+=(v2[i]*v3[i]);
    }
   (*cov)=sb*sb*(s0*s0+(xxx-mu)*(xxs-mu))*rho(xxx,xxs)
     -det3(m11,m12,v13,m12,m22,v23,m31,m32,v33)/det2;
   (*mean)=det3(m11,m12,p13,m12,m22,p23,m31,m32,p33)/det2;
  return;
}
```

update2.c – Standard model

```
#include "prob.h"

void PY_X0(double *mean, double *var)
{
  VECd v1,v2;
```

```
  int i;

  for(i=1;i<=n;i++)
    v1[i]=1;
  solve_y(&v,v1,v2);
  (*var)=1/(s0*s0);
  for(i=1;i<=n;i++)
    (*var)+=(b*b*v2[i]/sb/sb);
  (*var)=1/(*var);
  (*mean)=mu/(s0*s0);
  for(i=1;i<=n;i++)
    (*mean)-=(v2[i]*(y[i]-b*x[i])*b/sb/sb);
  (*mean)*=(*var);
  return;
}

void PY_B(double xxx, double xxs, double *mean, double *cov)
{
  VECd v1,v2,v3,v4;
  int i;
  double t1,t2;

  for(i=1;i<=n;i++)
    {
      v1[i]=(xxx-mu)*rho(xxx,xxs);
      v3[i]=(xxs-mu)*rho(xxx,xxs);
    }
  solve_y(&v,v1,v2);
  solve_y(&v,v3,v4);
  (*cov)=sb*sb*rho(xxx,xxs);
  t2=t1=0;
  for(i=1;i<=n;i++)
    {
      (*cov)-=(sb*sb*v1[i]*v4[i]);
      t1+=v2[i];
      t2+=v4[i];
    }
  (*cov)+=(t1*t2*s0s0_t*b*b);
  (*mean)=b;
  for(i=1;i<=n;i++)
    (*mean)+=(v2[i]*(y[i]-b*(x[i]-mu_t)));
}

void PY_Y(double xxx, double xxs, double *mean, double *cov)
{
  VECd v1,v2,v3,v4;
  int i;
  double t1,t2;
  for(i=1;i<=n;i++)
    {
      v1[i]=(s0*s0+(xxx-mu)*(x[i]-mu))*rho(xxx,x[i]);
      v3[i]=(s0*s0+(xxs-mu)*(x[i]-mu))*rho(xxs,x[i]);
    }
  solve_y(&v,v1,v2);
  solve_y(&v,v3,v4);
  (*cov)=(sb*sb*(s0*s0+(xxx-mu)*(xxs-mu))*rho(xxx,xxs));
  t1=1;t2=1;
  for(i=1;i<=n;i++)
    {
      (*cov)-=(sb*sb*v1[i]*v4[i]);
      t1-=v2[i];
      t2-=v4[i];
    }
  (*cov)+=(t1*t2*s0s0_t*b*b);
  (*mean)=b*(xxx-mu_t);
  for(i=1;i<=n;i++)
```

```
    (*mean)+=(v2[i]*(y[i]-b*(x[i]-mu_t)));
}
```

update3.c – Standard model – Newtonian estimates

```
#include "prob.h"

void PY_X0(double *mean, double *var)
{
  int i;

  *var=s0*s0*sb*sb/t;
  *mean=mu*sb*sb;
  for(i=1;i<=n;i++)
    *mean-=(s0*s0*b*c1[i]*(y[i]-b*x[i]));
  *mean/=t;
}

void PYV(VECD xxx, int len, double MU,
        VECD Bx, VECD Yx,
        MATD BBx, MATD YYx, MATD BYx,
        VECD X1x, VECD X2x, MATD XXx)
{
  MATD vc,vd;
  MATD vcx,vdx;
  VECd vt;
  VECD vc1,vd1;
  double texp;
  VECd temp1,temp2;
  int i,j,j1,j2;

  for(j=0;j<len;j++)
    {
      vc1[j]=1;vd1[j]=0;
    }
  for(i=1;i<=n;i++)
    {
      vt[i]=y[i]-b*x[i]+b*MU;
      for(j=0;j<len;j++)
        {
          vc1[j]-=(
                   (
                    vc[j][i]=(s0*s0+(xxx[j]-mu)*(x[i]-mu))*
                    rho(xxx[j],x[i])
                    )
                   *c1[i]);
          vd1[j]+=((vd[j][i]=(x[i]-mu)*rho(xxx[j],x[i]))*c1[i]);
        }
    }
  for(j=0;j<len;j++)
    {
      for(i=1;i<=n;i++)
        temp1[i]=vc[j][i];
      solve_y(&v,temp1,temp2);
      for(j1=1;j1<=n;j1++)
      for(i=1;i<=n;i++)
        {
          vcx[j][i]=temp2[i];
          temp1[i]=vd[j][i];
        }
      solve_y(&v,temp1,temp2);
      for(i=1;i<=n;i++)
```

```
        vdx[j][i]=temp2[i];
      Yx[j]=(Bx[j]=b)*(xxx[j]-MU);
      for(i=1;i<=n;i++)
        {
          Bx[j]+=(vdx[j][i]*vt[i]);
          Yx[j]+=(vcx[j][i]*vt[i]);
        }
    }

  for(j1=0;j1<len;j1++)
    {
    for(j2=0;j2<len;j2++)
      {
        texp=rho(xxx[j1],xxx[j2]);
        BBx[j1][j2]=texp+s0*s0*b*b*vd1[j1]*vd1[j2]/t;
        YYx[j1][j2]=texp*(s0*s0+(xxx[j1]-mu)*(xxx[j2]-mu))+s0*s0*b*b*vc1[j1]*vc1[j2]/t;
        BYx[j1][j2]=(xxx[j2]-mu)*texp+s0*s0*b*b*vd1[j1]*vc1[j2]/t;
        for(i=1;i<=n;i++)
          {
            BBx[j1][j2]-=(vd[j2][i]*vdx[j1][i]);
            YYx[j1][j2]-=(vc[j2][i]*vcx[j1][i]);
            BYx[j1][j2]-=(vc[j2][i]*vdx[j1][i]);
          }
        BBx[j1][j2]*=(sb*sb);
        YYx[j1][j2]*=(sb*sb);
        BYx[j1][j2]*=(sb*sb);
      }
    X1x[j1]=xxx[j1]-Yx[j1]/Bx[j1];
    X2x[j1]=xxx[j1]-(Yx[j1]*Bx[j1]*Bx[j1]+Yx[j1]*BBx[j1][j1]-Bx[j1]*BYx[j1][j1])/Bx[j1]/Bx[j1]/Bx[j1];
  }
  for(j1=0;j1<len;j1++)
    for(j2=0;j2<len;j2++)
        XXx[j1][j2]=(YYx[j1][j2]*Bx[j1]*Bx[j2]+BBx[j1][j2]*Yx[j1]*Yx[j2]
                     -BYx[j1][j2]*Yx[j1]*Bx[j2]-BYx[j2][j1]*Bx[j1]*Yx[j2])
          /Bx[j1]/Bx[j1]/Bx[j2]/Bx[j2];

  return;
}
```

update4.c – Extended model – Newtonian estimates

```
#include "prob.h"

void PY_X0(double *mean, double *var)
{
  int i;

  *var=s0*s0;
  *mean=mu;
}

double det3(double a11, double a21, double a31,
            double a12, double a22, double a32,
            double a13, double a23, double a33)
{
  return(a11*a22*a33+a21*a32*a13+a31*a12*a23-
         a11*a32*a23-a21*a12*a33-a31*a22*a13);
}

void PYV(VECD xxx, int len, double MU,
        VECD Bx, VECD Yx,
        MATD BBx, MATD YYx, MATD BYx,
```

```
        VECD X1x, VECD X2x, MATD XXx)
{
  VECd cx,cy;
  MATD cd,cc;
  MATD vc,vd;
  VECD cc1,ccx,dc1,dcx,ccy,dcy;
  MATD ccc,ccd,dcd;
  double tdet,texp;
  VECd temp1,temp2;
  int i,j,j1,j2;
  double t11,t1x,txx,t1y,txy;

  solve_y(&v,x,cx);
  solve_y(&v,y,cy);

  t11=t1x=txx=t1y=txy=0;
  for(i=1;i<=n;i++)
    {
      t11+=c1[i];
      t1x+=cx[i];
      t1y+=(c1[i]*y[i]);
      txx+=(cx[i]*x[i]);
      txy+=(cx[i]*y[i]);
    }
  tdet=t11*txx-t1x*t1x;

  for(j=0;j<len;j++)
    {
      cc1[j]=dc1[j]=ccx[j]=dcx[j]=0;
    }
  for(j=0;j<len;j++)
    {
      for(i=1;i<=n;i++)
        {
          cc1[j]+=(
                   (
                    temp1[i]=vc[j][i]=
                    (s0*s0+(xxx[j]-mu)*(x[i]-mu))*
                    rho(xxx[j],x[i])
                    )
                   *c1[i]);
          ccx[j]+=(vc[j][i]*cx[i]);
          ccy[j]+=(vc[j][i]*cy[i]);
        }
      solve_y(&v,temp1,temp2);
      for(i=1;i<=n;i++)
        {
          cc[j][i]=temp2[i];
          dc1[j]+=(
                  (
                   temp1[i]=vd[j][i]=
                   (x[i]-mu)*rho(xxx[j],x[i])
                   )
                  *c1[i]);
          dcx[j]+=(vd[j][i]*cx[i]);
          dcy[j]+=(vd[j][i]*cy[i]);
        }
      solve_y(&v,temp1,temp2);
      for(i=1;i<=n;i++)
        cd[j][i]=temp2[i];
    }

  for(j1=0;j1<len;j1++)
    for(j2=0;j2<len;j2++)
      {
        ccc[j1][j2]=ccd[j1][j2]=dcd[j1][j2]=0;
```

```
        for(i=1;i<=n;i++)
          {
            ccc[j1][j2]+=(cc[j1][i]*vc[j2][i]);
            ccd[j1][j2]+=(cc[j1][i]*vd[j2][i]);
            dcd[j1][j2]+=(cd[j1][i]*vd[j2][i]);
          }
      }
  b=(t11*txy-t1x*t1y)/tdet;
  for(j1=0;j1<len;j1++)
    {
      Yx[j1]=det3(t11,t1x,t1y,t1x,txx,txy,cc1[j1]-1,ccx[j1]-xxx[j1],ccy[j1])/tdet;
      Bx[j1]=det3(t11,t1x,t1y,t1x,txx,txy,dc1[j1],dcx[j1]-1,dcy[j1])/tdet;
      for(j2=0;j2<len;j2++)
        {
          texp=exp(-th*(xxx[j1]-xxx[j2])*(xxx[j1]-xxx[j2]));
          YYx[j1][j2]=sb*sb*texp*(s0*s0+(xxx[j1]-mu)*(xxx[j2]-mu))-
    sb*sb*det3(t11,t1x,cc1[j2]-1,t1x,txx,ccx[j2]-xxx[j2],cc1[j1]-1,ccx[j1]-xxx[j1],ccc[j1][j2])/tdet;
          BBx[j1][j2]=sb*sb*texp-
            sb*sb*det3(t11,t1x,dc1[j2],t1x,txx,dcx[j2]-1,dc1[j1],dcx[j1]-1,dcd[j1][j2])/tdet;
          BYx[j1][j2]=sb*sb*texp*(xxx[j2]-mu)-
            sb*sb*det3(t11,t1x,cc1[j2]-1,t1x,txx,ccx[j2]-xxx[j2],dc1[j1],dcx[j1]-1,ccd[j2][j1])/tdet;
        }
    }
  for(j1=0;j1<len;j1++)
    {
    X1x[j1]=xxx[j1]-Yx[j1]/Bx[j1];
    X2x[j1]=xxx[j1]-(Yx[j1]*Bx[j1]*Bx[j1]+Yx[j1]*BBx[j1][j1]-Bx[j1]*BYx[j1][j1])/Bx[j1]/Bx[j1]/Bx[j1];
      for(j2=0;j2<len;j2++)
        XXx[j1][j2]=(YYx[j1][j2]*Bx[j1]*Bx[j2]+BBx[j1][j2]*Yx[j1]*Yx[j2]
                    -BYx[j1][j2]*Yx[j1]*Bx[j2]-BYx[j2][j1]*Bx[j1]*Yx[j2])
                                    /Bx[j1]/Bx[j1]/Bx[j2]/Bx[j2];
    }
  return;
}
```

C.3.6 crit.c

Returns the value of the criterion, crit_no=1 is C_2, crit_no=3 is C_1, crit_no=5 gives the "inverse interpolation" method.

```
#include "prob.h"

double criterion(double xxx,int crit_no)
{
  double t1,t2;
  switch(crit_no)
    {
    case 1: PY_Y(xxx,xxx,&t1,&t2);
      return(-(t1*t1+t2)/Cov_Y_Y(xxx,xxx));
    case 2: PY_Y(xxx,xxx,&t1,&t2);
      return(-t1*t1/t2);
    case 3: PY_Y(xxx,xxx,&t1,&t2);
      return(-t1*t1-t2);
    case 4: PY_Y(xxx,xxx,&t1,&t2);
      return(-t1*t1/t2-log(t2));
    case 5: PY_Y(xxx,xxx,&t1,&t2);
      return(-t1*t1);
    default: PY_X0(&t1,&t2);
      return(t2/s0/s0);
```

```
    }
}
```

C.3.7 setup2.c, setup3.c and add_point2.c.

setup2.c and setup3.c read in the parameter values (th–θ, b–b, mu–μ_0, sb–σ_b, s0–σ_0), the criteria to be used crit_no, the test function fun_no, and the first and second order modification methods mu_no and var_no. setup2.c is for the standard model, and adds one point, whereas setup3.c is for the extended model, and adds two points at $\mu_0 \pm b$ (note that b is not needed in its true sense). add_point2.c adds the next point to the design grid and updates the variance matrix.

setup2.c – Standard model

```
#include "prob.h"

void setup(int *crit_no,int *mu_no, int *var_no, int *fun_no)
{
  n=1;
  scanf("%d %d %d %d %le %le %le %le %le",
        crit_no,mu_no,var_no,fun_no,
        &mu,&s0,&b,&sb,&th);
  v.m.m[1][1]=s0*s0;
  v.m.rows=v.m.columns=1;
  t=sb*sb+s0*s0*b*b*(c1[1]=1/v.m.m[1][1]);
  y[1]=f(xx[1]=x[1]=mu,*fun_no);
  return;
}
```

setup3.c – Extended model

```
#include "prob.h"

void setup(int *crit_no,int *mu_no, int *var_no, int *fun_no)
{
  int ttt;
  VECd temp;
  n=2;
  scanf("%d %d %d %d %le %le %le %le %le",
        crit_no,mu_no,var_no,fun_no,
        &mu,&s0,&b,&sb,&th);

  y[1]=f(xx[1]=x[1]=mu-b,*fun_no);
  y[2]=f(xx[2]=x[2]=mu+b,*fun_no);
  v.m.m[1][1]=v.m.m[2][2]=s0*s0+b*b;
  v.m.m[1][2]=v.m.m[2][1]=rho(-b,b)*(s0*s0-b*b);
  v.m.rows=v.m.columns=2;
```

```
  ttt=solve_0(&v);
  temp[1]=temp[2]=1;
  solve_y(&v,temp,c1);
  t=c1[1]+c1[2];
  return;
}
```

add_point2.c

```
#include "prob.h"

int add_x(double xxx,int fun_no)
{
  int ptr,l1;
  int ttt;
  VECd temp;

  if(xxx>xx[n])
    xx[n+1]=xxx;
  else
    {
      for(ptr=n;(ptr>0 && xxx<xx[ptr]);ptr--)
        xx[ptr+1]=xx[ptr];
      xx[ptr+1]=xxx;
    }
  v.m.rows=v.m.columns=++n;
  x[n]=xxx;
  y[n]=f(xxx,fun_no);
  for(l1=1;l1<n;l1++)
    v.m.m[n][l1]=v.m.m[l1][n]=(s0*s0+(xxx-mu)*(x[l1]-mu))*rho(xxx,x[l1]);
  v.m.m[n][n]=s0*s0+(xxx-mu)*(xxx-mu);
  for(l1=1;l1<=n;l1++)
    temp[l1]=1;
  ttt=solve_0(&v);
  solve_y(&v,temp,c1);
  if(!ttt)
    {
      t=0;
      for(l1=1;l1<=n;l1++)
        t+=c1[l1];
      t=sb*sb+b*b*s0*s0*t;
    }
  return(ttt);
}
```

C.3.8 file_print1.c and file_print2.c

These routines output the results to a file in rows of 5 for readability. file_print2.c just contains the routine file_print() which is also in file_print1.c.

```
#include "prob.h"

void file_print(FILE *fp,double value,int *out_count)
{
```

```
  fprintf(fp,"%15.7f ",value);
  if((++*out_count)==5)    {fprintf(fp,"\n");*out_count=0;}
}

void file_print_x(FILE *fp)
{
  double x_loop;
  int out_count;

  out_count=0;
  for(x_loop=-3;x_loop<=3.001;x_loop+=.02)
    file_print(fp,x_loop,&out_count);
  fprintf(fp,"\n");
}

void file_print_f(FILE *fp,int fun_no)
{
  double x_loop;
  int out_count;

  out_count=0;
  for(x_loop=-3;x_loop<=3.001;x_loop+=.02)
    file_print(fp,f(x_loop,fun_no),&out_count);
  fprintf(fp,"\n");
}

void file_print_crit(FILE *fp,int crit_no)
{
  double x_loop;
  int out_count;

  out_count=0;
  for(x_loop=-3;x_loop<=3.001;x_loop+=.02)
    file_print(fp,criterion(x_loop,crit_no),&out_count);
  fprintf(fp,"\n");
}

void file_print_y(FILE *fp)
{
  double x_loop,t1,t2;
  int out_count;

  out_count=0;
  for(x_loop=-3;x_loop<=3.001;x_loop+=.02)
    {
      PY_Y(x_loop,-9999,&t1,&t2);
      file_print(fp,t1,&out_count);
    }
  fprintf(fp,"\n");
}
```

C.3.9 compute1.c

This short routine computes the predictor of X_0, and if mu_no is non zero updates b by replacing it with its prevision at the current estimate of X_0.

```
#include "prob.h"
```

```
int compute_mean(int mu_no,int var_no,int crit_no)
{
  double t1,t2;
  int i,j;

  PY_X0(&mumu,&t1);
  mu_t=mumu;s0s0_t=t1;
  switch(mu_no)
    {
    case 2 : {mu=maximize_x(-3.0,3.0,PREC,crit_no);break;}
    case 1 : {mu=mumu;break;}
    default : {printf("%10.5f \n",mumu);break;}
    }
  if(mu_no!=0)
    {
      PY_B(mu,mu,&t1,&t2);
      b=t1;
      for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
          v.m.m[i][j]=(s0*s0+(x[i]-mu)*(x[j]-mu))*rho(x[i],x[j]);
      return(solve_0(&v));
      PY_X0(&mu_t,&s0s0_t);
    }
  return(0);
}
```

C.3.10 main1.c, main2.c, main3.c and main4.c

These are the main routines which tie all the other routines together, `main1.c` is for the extended model with variance modified criteria is, `main2.c` is for the standard model with variance modified criteria, `main3.c` is for the standard model using "Newtonian" estimates and `main4.c` is for the extended model using "Newtonian" estimates.

main1.c – extended model

```
#include "prob.h"

main(int argc,char *argv[])
{
  int loop,finish,error;
  double t1,t2,t3;
  int y_flag,crit_flag,mu_no,crit_no,fun_no,var_no;

  if (argc == 1)
    y_flag=crit_flag=0;
  else
    {
      switch((*++argv)[0])
        {
        case 'y' : y_flag=1;crit_flag=0;break;
        case 'c' : y_flag=0;crit_flag=1;break;
```

```
      case 'b' : y_flag=crit_flag=1;break;
      default  : y_flag=crit_flag=0;
      }
  }
--argc;
if (y_flag==1)
  {
    if(!(--argc) || (file_y = fopen(*++argv,"w")) == NULL)
      {
        fprintf(stderr,"OUTPUT FILE CREATION ERROR: %d\n",argc);
        exit(2);
      }
  }
if (crit_flag==1)
  {
    if(!(--argc) || (file_crit = fopen(*++argv,"w")) == NULL)
      {
        fprintf(stderr,"OUTPUT FILE CREATION ERROR : %d\n",argc);
        exit(2);
      }
  }
do
  {
    setup(&crit_no,&mu_no,&var_no,&fun_no);
    if(crit_no>=0)
      {
        printf("Criterion_no: %d Mean_adj_no: %d Var_adj_no: %d Function_no: %d\n",
                crit_no,         mu_no,          var_no,        fun_no);
        printf("Mu : %6.3f Si_0: %6.3f b: %6.3f Si_b: %6.3f Theta: %6.3f\n",mu,s0,b,sb,th);
        printf("        X_n          Y_n          Mu_0          b           Si_0         Si_b\n");
        if(y_flag)
          {
            file_print_x(file_y);
            file_print_f(file_y,fun_no);
          }
        if(crit_flag)
            file_print_x(file_crit);
        for(loop=2;loop<=MAX_SIZE-2;loop++)
          {
            if(error=compute_mean(mu_no,var_no,crit_no))
              break;
            printf("%2d %12.7f %12.7f %12.7f %12.7f %12.7f %12.7f\n",
                  n,x[n],y[n],mu,b,s0,sb);
            if(y_flag)
              file_print_y(file_y);
            if(crit_flag)
              file_print_crit(file_crit,crit_no);
            if(finish=(J_ABS(y[n])<END_CONDITION))
              break;
            else
              {
                if(mu_no==2)
                  error=2*add_x(mu,fun_no);
                else
                  error=3*add_x(maximize_x(-3.0,3.0,PREC,crit_no),fun_no);
              }
            if(error)
              break;
          }
        if(error)
          printf("Stopped due to singularity error\n");
        else
          {
            if(!finish)
              {
                if(error=4*compute_mean(mu_no,var_no,crit_no))
```

```
                    break;
                  printf("%2d %12.7f %12.7f %12.7f %12.7f %12.7f %12.7f \n",
                         n,x[n],y[n],mu,b,s0,sb);
                  if(y_flag)
                    file_print_y(file_y);
                  if(crit_flag)
                    file_print_crit(file_crit,crit_no);
                }
            }
        }
      if(error)
        {
          printf("Error code : %d\n",error);
          break;
        }
    }
  while(crit_no>=0);
}
```

main2.c – standard model

```
#include "prob.h"

main(int argc,char *argv[])
{
  int loop,finish,error;
  int y_flag,crit_flag,mu_no,crit_no,fun_no,var_no;

  if (argc == 1)
    y_flag=crit_flag=0;
  else
    {
      switch((*++argv)[0])
        {
        case 'y' : y_flag=1;crit_flag=0;break;
        case 'c' : y_flag=0;crit_flag=1;break;
        case 'b' : y_flag=crit_flag=1;break;
        default  : y_flag=crit_flag=0;
        }
    }
  --argc;
  if (y_flag==1)
    {
      if(!(--argc) || (file_y = fopen(*++argv,"w")) == NULL)
        {
          fprintf(stderr,"OUTPUT FILE CREATION ERROR: %d\n",argc);
          exit(2);
        }
    }
  if (crit_flag==1)
    {
      if(!(--argc) || (file_crit = fopen(*++argv,"w")) == NULL)
        {
          fprintf(stderr,"OUTPUT FILE CREATION ERROR : %d\n",argc);
          exit(2);
        }
    }
  do
    {
      setup(&crit_no,&mu_no,&var_no,&fun_no);
      if(crit_no>=0)
        {
```

```
        printf("Criterion_no: %d Mean_adj_no: %d Var_adj_no: %d Function_no: %d\n",
               crit_no,          mu_no,          var_no,        fun_no);
        printf("Mu : %6.3f Si_0: %6.3f b: %6.3f Si_b: %6.3f Theta: %6.3f\n",mu,s0,b,sb,th);
        printf("        X_n         Y_n          Mu_0          b           Si_0         Si_b\n");
        if(y_flag)
          {
            file_print_x(file_y);
            file_print_f(file_y,fun_no);
          }
        if(crit_flag)
            file_print_x(file_crit);
        for(loop=1;loop<=MAX_SIZE-2;loop++)
          {
            if(error=compute_mean(mu_no,var_no,crit_no))
              break;
            printf("%2d %12.7f %12.7f %12.7f %12.7f %12.7f %12.7f\n",
                   n,x[n],y[n],mu,b,s0,sb);
            if(y_flag)
              file_print_y(file_y);
            if(crit_flag)
              file_print_crit(file_crit,crit_no);
            if(finish=(J_ABS(y[n])<END_CONDITION))
              break;
            else
              {
                if(mu_no==2)
                  error=2*add_x(mu,fun_no);
                else
                  error=3*add_x(maximize_x(-3.0,3.0,PREC,crit_no),fun_no);
              }
            if(error)
              break;
          }
        if(error)
          printf("Stopped due to singularity error\n");
        else
          {
            if(!finish)
              {
                if(error=4*compute_mean(mu_no,var_no,crit_no))
                  break;
                printf("%2d %12.7f %12.7f %12.7f %12.7f %12.7f %12.7f \n",
                       n,x[n],y[n],mu,b,s0,sb);
                if(y_flag)
                  file_print_y(file_y);
                if(crit_flag)
                  file_print_crit(file_crit,crit_no);
              }
          }
      }
    if(error)
      {
        printf("Error code : %d\n",error);
        break;
      }
  }
while(crit_no>=0);
}
```

main3.c – standard model – Newtonian estimates

```
#include "prob.h"
```

```
FILE *fpred, *fcov;
main(int argc,char *argv[])
{
  int l1,l2;
  int out_count;
  int loop,finish,error,pos;
  double mustar,s0star,temp;
  int y_flag,crit_flag,mu_no,crit_no,fun_no,var_no;
  VECD xxxx,Bxx,Yxx,X1xx,X2xx;
  MATD BBxx,YYxx,BYxx,XXxx;

  if(argc<3)
    {
      fprintf(stderr,"INSUFFICIENT FILES\n");
      exit(2);
    }
  if((fpred = fopen(*++argv,"w")) == NULL)
    {
      fprintf(stderr,"OUTPUT FILE CREATION ERROR: %d\n",argc);
      exit(2);
    }
  if((fcov= fopen(*++argv,"w")) == NULL)
    {
      fprintf(stderr,"OUTPUT FILE CREATION ERROR : %d\n",argc);
      exit(2);
    }
  do
    {
      for(l1=0;l1<31;l1++)
        xxxx[l1]=l1*0.2-3;
      setup(&crit_no,&mu_no,&var_no,&fun_no);
      pos=mu*5.0+15.5;
      printf("%10.5f %10.5f\n",mu,xxxx[pos]);
      xxxx[pos]=mu;
      if(crit_no>=0)
        {
          printf("%12.7f %12.7f\n",mu,s0*s0);
          out_count=0;
          for(l1=0;l1<31;l1++)
            file_print(fpred,xxxx[l1],&out_count);
          out_count=0;
          for(l1=0;l1<31;l1++)
            file_print(fpred,f(xxxx[l1],fun_no),&out_count);
          out_count=0;
          for(l1=0;l1<31;l1++)
            file_print(fpred,(xxxx[l1]-mu)*b,&out_count);
          out_count=0;
          for(l1=0;l1<31;l1++)
            file_print(fpred,b,&out_count);
          out_count=0;
          for(l1=0;l1<31;l1++)
            for(l2=0;l2<31;l2++)
              file_print(fcov,Cov_B_B(xxxx[l1],xxxx[l2]),&out_count);
          out_count=0;
          for(l1=0;l1<31;l1++)
            for(l2=0;l2<31;l2++)
              file_print(fcov,Cov_B_Y(xxxx[l1],xxxx[l2]),&out_count);
          out_count=0;
          for(l1=0;l1<31;l1++)
            for(l2=0;l2<31;l2++)
              file_print(fcov,Cov_Y_Y(xxxx[l1],xxxx[l2]),&out_count);

          for(loop=0;loop<10;loop++)
            {
              PY_X0(&mustar,&s0star);
              PYV(xxxx,31,mustar,Bxx,Yxx,BBxx,YYxx,BYxx,X1xx,X2xx,XXxx);
```

```
            printf("%12.7f %12.7f %12.7f %12.7f %12.7f %12.7f\n",
                   mustar,sOstar,xxxx[pos],X1xx[pos],X2xx[pos],sqrt(XXxx[pos][pos]));
            out_count=0;
            for(l1=0;l1<31;l1++)
              file_print(fpred,Yxx[l1],&out_count);
            out_count=0;
            for(l1=0;l1<31;l1++)
              file_print(fpred,Bxx[l1],&out_count);
            out_count=0;
            for(l1=0;l1<31;l1++)
              file_print(fpred,X1xx[l1],&out_count);
            out_count=0;
            for(l1=0;l1<31;l1++)
              file_print(fpred,X2xx[l1],&out_count);
            out_count=0;
            for(l1=0;l1<31;l1++)
              file_print(fpred,sqrt(XXxx[l1][l1]),&out_count);
            out_count=0;
            for(l1=0;l1<31;l1++)
              for(l2=0;l2<31;l2++)
                file_print(fcov,BBxx[l1][l2],&out_count);
            out_count=0;
            for(l1=0;l1<31;l1++)
              for(l2=0;l2<31;l2++)
                file_print(fcov,BYxx[l1][l2],&out_count);
            out_count=0;
            for(l1=0;l1<31;l1++)
              for(l2=0;l2<31;l2++)
                file_print(fcov,YYxx[l1][l2],&out_count);
            out_count=0;
            for(l1=0;l1<31;l1++)
              for(l2=0;l2<31;l2++)
                file_print(fcov,XXxx[l1][l2],&out_count);
            if(crit_no==1)
              add_x((temp=X1xx[pos]),fun_no);
            else
              add_x((temp=X2xx[pos]),fun_no);
            pos=temp*5.0+15.5;
            xxxx[pos]=temp;
          }
        }
    }
  while(crit_no>=0);
}
```

main4.c – extended model – Newtonian estimates

```
#include "prob.h"
FILE *fpred, *fcov;
main(int argc,char *argv[])
{
  int l1,l2;
  int out_count;
  int loop,finish,error,pos;
  double mustar,sOstar,temp;
  int y_flag,crit_flag,mu_no,crit_no,fun_no,var_no;
  VECD xxxx,Bxx,Yxx,X1xx,X2xx;
  MATD BBxx,YYxx,BYxx,XXxx;

  if(argc<3)
    {
      fprintf(stderr,"INSUFFICIENT FILES\n");
```

```
    exit(2);
  }
if((fpred = fopen(*++argv,"w")) == NULL)
  {
    fprintf(stderr,"OUTPUT FILE CREATION ERROR: %d\n",argc);
    exit(2);
  }
if((fcov= fopen(*++argv,"w")) == NULL)
  {
    fprintf(stderr,"OUTPUT FILE CREATION ERROR : %d\n",argc);
    exit(2);
  }
do
  {
    setup(&crit_no,&mu_no,&var_no,&fun_no);
    for(l1=0;l1<31;l1++)
      xxxx[l1]=l1*0.2-3;
    pos=x[1]*10.0+15.5;
    xxxx[pos]=x[1];
    pos=x[2]*10.0+15.5;
    xxxx[pos]=x[2];
    if(crit_no>=0)
      {
        for(loop=0;loop<10;loop++)
          {
            PY_X0(&mustar,&s0star);
            PYV(xxxx,31,mustar,Bxx,Yxx,BBxx,YYxx,BYxx,X1xx,X2xx,XXxx);
            printf("%10.7f %10.7f %10.7f %10.7f %10.7f %10.7f %10.7f\n",
                   mustar,s0star,b,xxxx[pos],X1xx[pos],X2xx[pos],sqrt(XXxx[pos][pos]));
            out_count=0;
            for(l1=0;l1<31;l1++)
              file_print(fpred,Yxx[l1],&out_count);
            out_count=0;
            for(l1=0;l1<31;l1++)
              file_print(fpred,Bxx[l1],&out_count);
            out_count=0;
            for(l1=0;l1<31;l1++)
              file_print(fpred,X1xx[l1],&out_count);
            out_count=0;
            for(l1=0;l1<31;l1++)
              file_print(fpred,X2xx[l1],&out_count);
            out_count=0;
            for(l1=0;l1<31;l1++)
              file_print(fpred,sqrt(XXxx[l1][l1]),&out_count);
            out_count=0;
            for(l1=0;l1<31;l1++)
              for(l2=0;l2<31;l2++)
                file_print(fcov,BBxx[l1][l2],&out_count);
            out_count=0;
            for(l1=0;l1<31;l1++)
              for(l2=0;l2<31;l2++)
                file_print(fcov,BYxx[l1][l2],&out_count);
            out_count=0;
            for(l1=0;l1<31;l1++)
              for(l2=0;l2<31;l2++)
                file_print(fcov,YYxx[l1][l2],&out_count);
            out_count=0;
            for(l1=0;l1<31;l1++)
              for(l2=0;l2<31;l2++)
                file_print(fcov,XXxx[l1][l2],&out_count);
            if(crit_no==1)
              add_x((temp=X1xx[pos]),fun_no);
            else
              add_x((temp=X2xx[pos]),fun_no);
            pos=temp*5.0+15.5;
            xxxx[pos]=temp;
```

```
        }
      }
    }
  while(crit_no>=0);
}
```

C.3.11 makefile

As a final subsection, we include the *makefile* which links all these subroutines together.

```
MAIN1_OBJECTS=file_print1.o fun1.o mat2.o cov1.o update1.o setup3.o add_point2.o rho1.o crit1.o
golden.o compute1.o main1.c
MAIN2_OBJECTS=file_print1.o fun1.o mat2.o cov1.o update2.o setup2.o add_point2.o rho1.o crit1.o
golden.o compute1.o main2.c
MAIN3_OBJECTS=file_print2.o fun1.o mat2.o cov1.o update3.o setup2.o add_point2.o rho1.o main3.c
MAIN4_OBJECTS=file_print2.o fun1.o mat2.o cov1.o update4.o setup3.o add_point2.o rho1.o main4.c
ALL_OBJECTS=add_point2.o compute1.o cov1.o crit1.o file_print1.o file_print2.o fun1.o golden.o mat2.o
rho1.o rho2.o rho3.o rho4.o setup2.o setup3.o update1.o update2.o update3.o update4.o
ALL_SOURCES=main1.c main2.c main3.c main4.c
FLAGS=-O -W
INC=prob.h
all:  main4 main3 main2 main1
##############################################################################
# main4 is the main file for the 'Newtonian' estimate on extended model #
##############################################################################
main4:          $(MAIN4_OBJECTS) $(INC)
                        gcc $(FLAGS) $(MAIN4_OBJECTS) -o main4 -lm
##############################################################################
# main3 is the main file for the 'Newtonian' estimate on standard model #
##############################################################################
main3:          $(MAIN3_OBJECTS) $(INC)
                        gcc $(FLAGS) $(MAIN3_OBJECTS) -o main3 -lm
##############################################################################
# main2 is the main file for the 2nd order mod crit on standard model   #
##############################################################################
main2:          $(MAIN2_OBJECTS) $(INC)
                        gcc $(FLAGS) $(MAIN2_OBJECTS) -o main2 -lm
##############################################################################
# main1 is the main file for the 2nd order mod crit on extended model   #
##############################################################################
main1:          $(MAIN1_OBJECTS) $(INC)
                        gcc $(FLAGS) $(MAIN1_OBJECTS) -o main1 -lm

add_point2.o:   add_point2.c $(INC)
                        gcc $(FLAGS) -c add_point2.c -o add_point2.o
compute1.o:     compute1.c $(INC)
                        gcc $(FLAGS) -c compute1.c -o compute1.o
cov1.o:         cov1.c $(INC)
                        gcc $(FLAGS) -c cov1.c -o cov1.o
crit1.o:        crit1.c $(INC)
                        gcc $(FLAGS) -c crit1.c -o crit1.o
file_print1.o:  file_print1.c $(INC)
                        gcc $(FLAGS) -c file_print1.c -o file_print1.o
file_print2.o:  file_print2.c $(INC)
                        gcc $(FLAGS) -c file_print2.c -o file_print2.o
fun1.o:         fun1.c $(INC)
                        gcc $(FLAGS) -c fun1.c -o fun1.o
golden.o:       golden.c $(INC)
```

```
                gcc $(FLAGS) -c golden.c -o golden.o
mat2.o:         mat2.c $(INC)
                gcc $(FLAGS) -c mat2.c -o mat2.o
rho1.o:         rho1.c $(INC)
                gcc $(FLAGS) -c rho1.c -o rho1.o
rho2.o:         rho2.c $(INC)
                gcc $(FLAGS) -c rho2.c -o rho2.o
rho3.o:         rho3.c $(INC)
                gcc $(FLAGS) -c rho3.c -o rho3.o
rho4.o:         rho4.c $(INC)
                gcc $(FLAGS) -c rho4.c -o rho4.o
setup2.o:       setup2.c $(INC)
                gcc $(FLAGS) -c setup2.c -o setup2.o
setup3.o:       setup3.c $(INC)
                gcc $(FLAGS) -c setup3.c -o setup3.o
update1.o:      update1.c $(INC)
                gcc $(FLAGS) -c update1.c -o update1.o
update2.o:      update2.c $(INC)
                gcc $(FLAGS) -c update2.c -o update2.o
update3.o:      update3.c $(INC)
                gcc $(FLAGS) -c update3.c -o update3.o
update4.o:      update4.c $(INC)
                gcc $(FLAGS) -c update4.c -o update4.o
```

Bibliography

[1975] BLIGHT, B. AND OTT, L. A bayesian approach to model inadequacy, for polynomial regression. *Biometrica*, **62**, 79-88.

[1991] CURRIN, C., MITCHELL, T., MORRIS, M. AND YLVISAKER, D. Bayesian prediction of deterministic function with applications to the design and analysis of computer experiments. *J. American Stat. Assoc.*, **86**, 953-963.

[1988] DIACONIS, P. Bayesian numerical analysis. *Statistical Decision Theory and Related Topics IV (S.S. Gupta and J. Berger, eds.)*. Springer-Verlang, New York. **1**, 163–175.

[1970] DE FINETTI, B. Theory of Probability Vol. 1. Wiley, (Translated into English 1974).

[1981] GOLDSTEIN, M. Revising previsions: A geometric interpretation. *J. Roy. Statist. Soc. B*, **43**, 105–130.

[1983] GOLDSTEIN, M. Temporal Coherence. *Bayesian Statistics 2 (J.M. Bernado, M.H. DeGroot, D.V. Lindley, A.F.M. Smith, eds.)*

[1986] GOLDSTEIN, M. Separating Beliefs. *Essays in Honour of Bruno de Finetti: Bayesian Inference and decision Techniques (P.K. Goel & A. Zellner, eds.)* 197-215.

[1987] GOLDSTEIN, M. Systematic Analysis of Limited Belief Specifications. *The Statistician* **36**, 191–199.

[1988a] GOLDSTEIN, M. The data trajectory. *Bayesian Statistics 3 (J.M. Bernado, M.H. DeGroot, D.V. Lindley, A.F.M. Smith, eds.)* 189-209.

[1988b] GOLDSTEIN, M. Adjusting belief structures. *J. Roy. Statist. Soc. B*, **50**, 133–154.

[1991] GOLDSTEIN, M. Belief transforms and the comparison of hypothesis. *The Annals of Statistics,* **19**, 2067–2089.

[1985] KEE, R., GRCAR, J., SMOOKE, M. AND MILLER, J. A FORTRAN program for modeling steady laminar one-dimensional premixed flames. Sandia Report SAND85-8240 available from the National Technical Information Service, Springfield, Va 22161.

[1956] LINDLEY, D.V. On a measure of the information provided by an experiment. *Ann. Math. Statist.,* **27**, 986-1005.

[1989] MÖCKUS, J. *Bayesian approach to global optimization.* Mathematics and its applications (Soviet Series) Kluwer Academic Publishers

[1979] MOUCHART, M. AND SIMAR, L. Least square approximation in Bayesian analysis. *Bayesian Statistics,* Proceedings of the First Valencia International Meeting, 28 May - 2 June, 1979, 207–222.

[1984] NASSIF, S.R., STROJWAS, A.J. AND DIRECTOR, S.W. FABRICS II:A statistically based IC fabrication process simulator. *IEEE Trans. Computer-Aided Design*, **CAD-3**, 40-46.

[1978] O'HAGAN, A. Curve fitting and optimal design for prediction (with discussion). *J. Roy. Statist. Soc. B*, **40**, 1–42.

[1990] O'HAGAN, A. Bayesian intergration. *Technical Report 177,* University of Warwick.

[1991] O'HAGAN, A. Some bayesian numerical analysis. Fourth Valencia International Meeting on Bayesian Statistics, 15–20 April, 1991.

[1896] POINCARÉ, H. *Calcul des Probabilities,* Gauther-Villars

[1988] SACKS, J. AND SCHILLER, S. Spatial designs. *Statistical Decision Theory and Related Topics IV (S.S. Gupta and J. Berger, eds.).* **2** 385–399.

[1989a] SACKS, J., SCHILLER, S. AND WELCH, W. Design for Computer Experiments *Technometrics,* **31** 41-47.

[1989b] SACKS, J., WELCH, W., MITCHELL, T. AND WYNN, H. Design and analysis of computer experiments. *Statistical Science,* **4**, 409–435.

[1979] SCHAGEN, I.P. Interpolation in two dimensions — a new technique. *J. Inst. Maths Applics,* **23**, 53–59.

[1980a] SCHAGEN, I.P. Stochastic interpolating functions —applications in optimization. *J. Inst. Maths Applics,* **26**, 93–101.

[1980b] SCHAGEN, I.P. The use of stochastic processes in interpolation and approximation. *Int. J. Comput. Math. B,* **8**, 63–76.

[1984] SCHAGEN, I.P. Sequential exploration of unknown multi-dimensional functions as an aid to optimization. *IMA J. Num. Anal.* **4**, 337–346.

[1985] SILVERMAN, S. Some aspects of the spline smoothing approach to non-parametric curve fitting. *J. Roy. Statist. Soc. B*, **47**, 1-52.

[1978] WAHBA, G. Improper priors, spline smoothing and the problem of guarding against model errors in regression. *J. Roy. Statist. Soc. B*, **40**, 364–372.

[1992] WELCH, W., BUCK, R., SACKS, J., WYNN, H., MITCHELL, T. AND MORRIS, M. Screening, predicting and computer experiments. *Technometrics*, **34**, 15–25.

[1987] YLVISAKER, D. Prediction and design. *Annals of Statistics,* **15**, 1–19.