"Thinking inside the box": using derivatives to improve Bayesian black box emulation of computer simulators with application to compart mental models

Matthew R. H. Killeya

"Thinking Inside The Box"

Using Derivatives To Improve Bayesian Black Box Emulation Of Computer Simulators With Application To Compartmental Models.

Matthew R. H. Killeya

A Thesis presented for the degree of
Doctor of Philosophy

Statistics and Probability Group
Department of Mathematical Sciences
University of Durham
England

October 2004

Thinking Inside the Box: Using Derivatives to Improve Bayesian Black Box Emulation of Computer Simulators with Application to Compartmental Models.

Matthew R. H. Killeya

Submitted for the degree of Doctor of Philosophy
October 2004

Abstract

Increasingly, science relies on complex numerical models to aid understanding of physical phenomena. Often the equations in such models contain a high number of poorly known parameters so that the resulting output encodes much uncertainty. A 'computer simulator', which comprises the model equations together with a solver routine, produces a solution for a given choice of these 'input' parameters.

In cases where the dimension of the input parameter space is high, we can only hope to obtain a thin coverage of the space by running the simulator. Building a representation of the simulator output as a function of the input, then, is a statistical problem in which we observe output at a collection of input choices and, based on these observations, infer output values for unseen inputs about which we are uncertain. In a Bayesian context, this representation, termed the 'emulator', encodes our beliefs about the relationships between inputs and outputs.

Our interest is in exploiting the structure of compartmental models to aid in this process. Compartmental models are widely applied to model systems in the absence of fundamental equations to describe the processes of interest. We show that the structure of such models enables us to efficiently generate additional function information, in the form of input derivatives, each time we run the simulator and we adapt the emulator methodology to allow for derivatives. We show that considering derivatives offers a range of natural ways to aid assessment of prior beliefs and

that updating based on derivatives can lead to substantial reduction in emulator uncertainty. We show that, in addition, the model structure allows us to derive estimates of increased costs of generating derivatives which we can compare against the corresponding reduction in uncertainties.

We are motivated throughout by the problem of calibrating a compartmental model of plankton cycles at multiple locations in the sea, and we show that a knock-on effect of reduction of uncertainty by derivatives is an improvement in our ability to perform this calibration. The search for a model which could accurately reproduce plankton cycles at various physical locations, if successful, is thought to have significant ramifications for understanding climate change.

Declaration

The work in this thesis is based on research carried out at the Statistics and Probability Group, the Department of Mathematical Sciences, the University of Durham, England. No part of this thesis has been submitted elsewhere for any other degree or qualification and it all my own work unless referenced to the contrary in the text.

Acknowledgements

"The saying 'Having a Killey's thirst' records a man with a prodigious liquid capacity." (In "Surnames of the Manks" by Leslie Quilliam)

Thanks firstly to all those who came to the Queen Vic last night - under slightly false pretences - to celebrate the submission of my thesis. I wish I could remember more of the evening.

To my supervisor, Michael, for his constant support, insight, good humour and enthusiasm; the extent of which is such that he was able to convince me that I would rather go to Milton Keynes than Chile to present my work. Jonty for encouragement and enthusiasm and for help many times with R. Peter Craig for good humour and interesting conversations and for allowing me to fill his hard-drive with plankton data.

Of my friends, Giblin arrived in Durham at the same time as me and was the only one to stay the distance. I'll never forget that argument about sick in the bath that turned out to be our housemate's Body Shop 'body scrub with marmalade pieces'. Thanks also for Kenya; the only time I have ever sat next to a goat on public transport.

Harry the Belgian, thanks for our trip to Malta in the summer of second year and for arranging the accommodation; those three days on the streets at the start were an experience. Falling asleep with mild fatigue on the wall of the town square to awake and find a 200 foot drop the other side was the catalyst for celebrity status in small town Malta. Something everyone should experience.

Thanks also to Ibison for making my final year as an undergraduate a true pleasure.

Mohammed al Masri has put up with me at his place for the last two weeks and has been a truly great friend for the last two years. Our conversations have influenced many of my ideas and attitudes; whenever I am writing in future, I am sure I will pause many times and think 'What would Mohammed say?'

To Anton, whose unique brand of 'being a miserable bastard' had a cheering effect on me throughout the three years. Also to Lisa, who played a crucial part in the final stages in keeping me in music after mum had taken my entire CD collection home. To Harald and his housemates for providing me with food and shelter for the first two weeks in October when otherwise I would have been forced to sleep in the coffee room. Also thanks to Maha for several interesting conversations about the middle east.

To Frank Coolen who was responsible for convincing me to do a PhD in the first place. When I finally decided to do it, three days before graduation, he jumped up, said 'I have to make some phone calls' and ran out of Michael's office. The deadline for me to decide was the previous February. Thanks for persevering.

To Sharry, for Friday football and for going to extreme lengths - including an alleged on-the-spot fine from the police - in order to provide amusement on departmental outings. To Rachel Duke for never tiring of winding me up by email. And Michael McNamara; the only person I know to have fitted the phrases 'Jock Stein', 'Brandenburg gate' and 'Bradford and Bingley' into a single sentence.

Also from the department for making me laugh, helping me out or taking an interest in what I was up to (or in some cases, all three): Patrick Dorey, Vernon Armitage, James Gray, Probert, Wee Stevie, Debbie, Emily, Majid, Wadey, Mike, Owen, Ian Vernon, David Wooff, Gerda Arts, John Coleman, Bob Johnson and Clifford Johnson.

Thanks to John Hemmings in Southampton for always being helpful and quick to respond to oceanography-related queries. Stefano Conti from the University of Sheffield for useful conversations.

Thanks to Chris, Vanja and Stephen and to anyone who ever came to our house for food or *Risk*: Chris Prior, Beki, Frank, Pierpaolo, Steve Cox, Esti, Kate and Duncan, Harry Byrd, Nick-Pete-and-Griff amongst others. Thanks especially to

Shahram who never failed to turn up with a smile and a pizza from his shop (even when he was coming for a dinner party).

Thanks also to PC Jeff Barksby who had at least as much fun as we did at anti-war and other protests and who still enthusiastically asks me when the next one will be whenever I bump into him.

So many people at durham21: Mike, a brilliant deputy in my year as editor; Cat Evans, for bright and funny conversations; and Barney, our road trip to the Guardian Media Awards is one of the best days out I have had.

To my parents for all their love and support. You always made me feel that you were very proud of me, which is an amazing gift to give someone. To my brother, Adam, for those three-hour phone conversations from Japan which, when Mum and Dad asked what we talked about, neither of us could ever remember.

Finally, to Charlotte. A truly amazing person who makes me laugh everyday.

Contents

Abstract ii

Declaration iv

Acknowledgements v

1 Introduction 1

2 Calibrating the plankton cycle 6

2.1 Motivating problem 6

2.2 The model 9

2.3 Equations for the PZN simulator 13

2.3.1 Phytoplankton 13

2.3.2 Zooplankton 15

2.3.3 Nutrient 15

2.4 Simulator input: invariant input parameters 15

2.5 Simulator input: physical forcing 16

2.5.1 Mixed Layer Depth 17

2.5.2 Photosynthetically Available Radiation 17

2.5.3 Day length 20

2.6 System observations 20

2.6.1 Chlorophyll Data 20

2.6.2 Winter time Nutrient data 24

2.7 Recap: General features of the problem 26

3 Computer Simulators and Emulation **27**

3.1 Computer Simulators 27

3.1.1 Uncertainty on inputs 28

3.1.2 Model discrepancy 28

3.1.3 Observation error 29

3.1.4 Simulator uncertainty 30

3.2 Emulators 31

3.3 Bayesian approach 35

3.4 Partial Beliefs: Bayes Linear Methods 37

3.5 Bayes Linear emulators 40

4 Adjusting emulators based on derivatives **41**

4.1 Random Processes and derivatives 41

4.2 Beliefs about Derivatives 43

4.3 Discussion 47

4.4 Theoretical calculations of resolved variance 48

4.4.1 Variance resolved by a single input point 49

4.4.2 Several independent design points 52

4.5 Estimating Θ covariance parameters 54

4.6 Choice of Gaussian covariance function 57

5 Taking derivatives in compartmental simulators **59**

5.1 Derivatives and compartmental structure 59

5.2 Differentiating Euler's Rule 61

5.3 Higher order generalisation and adaptive stepsize 61

5.4 Exploiting compartmental structure 63

5.5 Implementation in Maple and R 65

5.6 Additional cost of derivatives: a heuristic 67

5.6.1 Derivation of heuristic 67

5.6.2 Discussion of heuristic assumptions 69

5.7 Localisation of inputs in the PZN model 70

6 Emulator construction and refinement for the plankton model 73

6.1 Choice of stations to emulate 73

6.2 Parameterisation of model output 77

6.3 Preliminary analysis at station 1115 79

6.3.1 Choice of active inputs 82

6.3.2 Assessing global prior mean and covariance parameters 87

6.3.3 Assessing variance parameters for ϵ and δ 88

6.3.4 Design of simulator runs 92

6.4 From exploratory analysis to prior emulator 94

6.5 Resolving emulator uncertainty at a single site 96

6.6 Simulator Diagnostics 100

6.7 From single to multiple sites 102

7 Calibrating the model 105

7.1 Linking simulator and physical system 105

7.2 Implausibility in a single output dimension 107

7.3 Combining Implausibilities 108

7.4 Bayesian vs Bayes Linear calibration 108

7.5 Choosing σ_e and σ_η 109

7.6 Preliminary comparison with and without derivatives 110

7.6.1 Function and derivative calibration trade-off 116

7.7 Calibration under maximal information 117

7.7.1 Lower dimensional projections of implausibility 117

7.7.2 'Most plausible' output 122

7.7.3 Refocusing and widening the scope of the calibration 126

7.7.4 Discussion in relation to Hemmings analysis 127

8 Conclusions and Future Directions 129

Appendix 137

A Additional Chapter 6 analysis 137

A.1 Derivative plots 138

A.2 Simulator Diagnostics 141

B PZN simulator run code with derivatives **144**

C Emulator updating code **159**

D Implausibility code **170**

E Miscellaneous code **173**

F Bessel functions **181**

G High order derivatives of the Gaussian covariance kernel **182**

List of Figures

2.1 Map showing station locations, taken from Hemmings et al. (2004). The subset of stations marked in black is used throughout this chapter to illustrate a range of physical observations and location-specific simulator data. 7

2.2 Schematic overview of the simulator. 10

2.3 Simulator output for the default input setting plotted against time (days) at the four demonstration stations. The first row corresponds to phytoplankton, $P(t)$, the second row to zooplankton, $Z(t)$, and the third row to nutrient, $N(t)$. 12

2.4 Schematic diagram of the simulator. 13

2.5 Mixed layer depth (m) against time (days) at the four stations. 18

2.6 Photosynthetically Available Radiation against time (days) at the four stations. 19

2.7 Observed chlorophyll (mmol m^{-3}) at the four locations recorded for 1998 where $t = 0$ corresponds to midnight GMT on 31st December 1997. 21

2.8 Optical interactions for Ocean Remote Sensing. Reproduced with permission from Andreas Neumann and Jill Schwarz. 23

2.9 Estimated annual maximum nutrient concentration in the mixed layer (mmol m^{-3}). Circles denote stations and additional crosses mark stations where nitrate estimates are available. 24

3.1 Conditional Independence Graph for the simulator, $s(\cdot)$, the best input, x_0, the physical system, y, and observations of the system, $\hat{y}$. Any two nodes are independent given the values of all their parent nodes. 29

3.2 Realisations from a Gaussian process with Matérn Correlation function, $R_{\nu,\theta}(x)$, shown for different ν and θ values. The final row shows $R_{\nu,\theta}(h)$ with θ fixed at the corresponding column value and varying $\nu = 1$ (black), $\nu = 4$ (red), $\nu = \infty$ (green). 33

4.1 (Top left) Covariance between function values and derivatives for the Gaussian covariance kernel in one dimension with hyper-parameters $\sigma_\epsilon = 1$, $\theta = 1.5$ plotted against the distance between observations, $h = ||x - x'||$. The thick lines shows $\text{Cov}[\epsilon(x), \epsilon(x')]$, the dashed line shows $\text{Cov}[\epsilon(x), d\epsilon(x')/dx']$ and the dotted line shows $\text{Cov}[d\epsilon(x)/dx, d\epsilon(x')/dx']$. The top right and bottom left plots show the constraining effects of function values and additionally with derivatives respectively. 45

4.2 Profile likelihood for θ for ten samples from a Gaussian Process with $r(h) = \exp^{-7.5h^2}$, each sample taken at seven points, without derivatives (top) and with derivatives (bottom). 56

5.1 Input and outputs appearing on each additive arc of the model. 71

6.1 Top: Euclidean distances of mixed layer depth (MLD) forcing functions (left) and photosynthetically available radiation (PAR) forcing functions (right) between pairs of stations. Light colours correspond to large distances and dark colours to small distances. Bottom: Mean and range of forcing function values at each station, again with MLD shown left and PAR shown right; circles denote mean values and lines connect minimum and maximum values and, in the MLD plot, colours denote groupings suggested by the MLD distances. 74

6.2 Map of stations shown with coloured outlines corresponding to the groupings identified by the mixed layer depth distances in Figure 6.1. The four stations marked black are those which we propose to emulate in this Chapter and then calibrate in Chapter 7. Station 1115, marked dark grey, is used to perform an exploratory analysis before we begin the emulation process. 76

6.3 Observed chlorophyll (mmol m^{-3}) at the four calibration locations recorded for 1998 where $t = 0$ corresponds to midnight GMT on 31st December 1997. Vertical lines determine intervals used for parameterisation as described in Section 6.2. 78

6.4 Station 1115: time series simulator output at different inputs for phytoplankton, $P(t)$, and nutrient, $N(t)$, plotted against time, t. 80

6.5 Station 1115: mean ± 2 s.d. phytoplankton simulator data. Black lines denote values for daily time series data and grey lines and rectangles denote values for each output of the interval re-parameterisation. 81

6.6 Station 1115: value of the largest and second largest phytoplankton input derivative mean and standard deviation, plotted at every seventh day of the original untransformed time series. The plot symbols are numbers corresponding to the input whose derivative is plotted and lines join values of inputs whose derivative appears at successive time points. For the mean plot, grey filled circles corresponds to derivatives whose values are negative. 83

6.7 Station 1115: Derivatives of re-parameterised outputs with respect to each input computed at the twenty five input values. The set of active variables, $x^* = (x_1, x_4, x_6, x_{10})$ for $\bar{P}_{1115}$, and $x^* = (x_1, x_4, x_6, x_{13})$ for N_{1115}, are coloured black and grey vertical lines denote ±2s.d. for each sample of derivatives. 84

6.8 Station 1115: simulator output, $\bar{P}_i$ and partial derivatives, $\partial \bar{P}_i / \partial x_1$, $i = 1, \ldots, 5$, plotted against x_1 for the first five plots, and simulator output, N_w, and partial derivative, $\partial N_w / \partial x_{13}$, plotted against x_{13} for the final plot. For each output, function values are marked by their run number and derivatives by the gradient of the line passing through the corresponding point. 85

6.9 Mean ±2 standard deviation of $C_1, \ldots, C_{13}$ values based on 25 minimisations of $wC_5 + (1 - w)C_{13}$, with each minimisation over 10,000 randomly generated 13-dimensional 200-point hypercubes. Results for different values of $w \in (0, 0.25, 0.5, 0.75, 1)$ are shown from left to right for each C_i. . 93

6.10 Station 1113: output parameterisation (top left) and derivatives of the re-parameterised outputs with respect to each input computed at the twenty five input values (remaining plots). See caption in Figure 6.7 for full description. 95

6.11 Adjusted standard deviation against number of simulator runs with derivatives (black) and without derivatives (grey) for outputs at Station 1113. Solid and dashed lines correspond respectively to the mean and maximum standard deviation taken over the input space. The light grey rectangle is defined by the horizontal lines $\sqrt{\sigma_\epsilon^2 + \sigma_\delta^2}$ and σ_δ, shown for reference. . . . 97

6.12 Station 1113: Separate components of adjusted standard deviation with derivatives (black) and without derivatives (grey). Solid lines correspond to $\mathrm{SD}_{D_n}[Bg(x)]$ and dashed lines to $\mathrm{SD}_{D_n}[\epsilon(x)]$. The light grey rectangle is defined by the horizontal lines $\sqrt{\sigma_\epsilon^2 + \sigma_\delta^2}$ and σ_δ, shown for reference. . . 98

6.13 Forecast diagnostics based on function values for the outputs at station 1113 plotted against the number of runs, n. Filled circles correspond to $s(x_{n+1})$, the value observed when running the simulator at x_{n+1}. Unfilled circles correspond to $\mathrm{E}_{D_n}[s(x_{n+1})]$ and vertical lines show $\pm 3\mathrm{SD}_{D_n}[s(x_{n+1})]$ for $D_n = S_n$. 101

6.14 Forecast diagnostics based on function values and derivatives for the outputs at station 1113 plotted against the number of runs, n. See Figure 6.13 for details ($D_n = (S_n, \nabla_{x^*} S_n)$). 102

6.15 Plots of $\Delta_{S_n} - \Delta_{(S_n, \nabla_{x^*} S_n)}$ for station 1113 (See Eqn. 6.12). The red dotted line shows Derr=0 and the blue line gives the running average of the first n Derr scores. 103

7.1 Proportion of input space ruled out by the set, $\bar{P}$, of phytoplankton outputs across all four stations with derivatives (black) and without derivatives (grey) plotted against the size n of the input set X_n at each station (so that e.g. $n = 10$ corresponds to ten runs at each station, and so forty in total). The numeric plotting symbols, $m = 0, 2$, correspond to $\sigma_\eta = m\sigma_e$, $m = 0, 2$. 111

7.2 Proportion of input space ruled out by the set, $\bar{P}_s$, of phytoplankton output for each station, s, with derivatives (black) and without derivatives (grey). 111

7.3 Proportion of points 'falsely' ruled out by phytoplankton outputs at station 1116 (top) and all four stations (bottom). The proportions are the number of points which are ruled out based on n input points but ruled in based on $n = 50$ points. Shown in each plot are the proportions with derivatives (black) and without derivatives (grey) for $\sigma_\eta = 0$ (thick line) and $\sigma_\eta = 2\sigma_e$ (dotted line). The cut off value in each plot is the implausibility score above which we rule points out. 113

7.4 Proportion of input space ruled out by the set, N_w, of nutrient outputs across all four stations with derivatives (black) and without derivatives (grey). See Figure 7.4 for more details. 114

7.5 Proportion of input space ruled out by the nutrient output, $N_{w,s}$, at each station, s, with derivatives (black) and without derivatives (grey). 114

7.6 $I_{\bar{P}}(x_1, x_4)$ (left) and $I_{\bar{P}}(x_6, x_{10})$ (right), shown after $n = 50$ runs at each station. Implausible areas are shaded darker, black dots denote input points at which simulator output has been observed at one of the four stations, and numbered circles correspond to the three most plausible input points. 119

7.7 $I_{\bar{P}_s}(x_1, x_4)$ (top) and $I_{\bar{P}_s}(x_6, x_{10})$ (bottom): Implausibility maximised over phytoplankton outputs and projected into two input dimensions, for each of the four calibration stations, shown after $n = 50$ runs. 119

7.8 'Four dimensional' plot of the top 1% 'most plausible' input combinations, all of which fall in either region 1 (left) or region 2 (right) of the (x_1, x_4) space. Outer grid lines correspond to the x_1 and x_4 dimensions with x_6 and x_{10} dimensions plotted within these grid lines, each on the range $[-1, 1]$. 120

7.9 Subregion of two dimensional $I_{\bar{P}}$ projection in Figure 7.6, with implausibility minimised over the subregions only. 120

7.10 $I_N(x_i, x_{13})$ for $i = 1, 4, 6$: Implausibility maximised over nutrient outputs and stations and projected into two input dimensions by minimising over remaining input dimensions, shown after $n = 50$ runs at each station. . . . 121

7.11 $I_{N_s}(x_i, x_{13})$ for $i = 1, 4, 6$: Implausibility based on the nutrient output at each of the calibration stations, projected into two input dimensions after $n = 50$ runs. 121

7.12 Simulator output at $\hat{x}_0$ (unfilled circles, with lines denoting $\pm 3\sigma_\eta$) and historical data (filled circles, with lines denoting $\pm 3\sigma_e$) for phytoplankton at the calibration stations. The dotted line gives the original, unparameterised phytoplankton output at $\hat{x}_0$ and the grey polygon shows the range spanned by $\pm 35\%$ of the original historical chlorophyll observations, as a guide to ± 3 measurement error for the calibration and validation stations. 123

7.13 Simulator nutrient output at $\hat{x}_0$ (unfilled circles, with lines denoting $\pm 3\sigma_\eta$) and historical nutrient data (filled circles, with lines denoting $\pm 3\sigma_e$) at the calibration and validation stations. Validation stations are shown with $\sigma_e = \sigma_\eta = 1$ as a guide only. 124

A.1 Station 1015 138

A.2 Station 1116 139

A.3 Station 1215 140

A.4 Station 1015 141

A.5 Station 1116 142

A.6 Station 1215 143

List of Tables

2.1 Inputs with Expert's Lower Bound, L, Default Value, D, and Upper Bound, U. 11

2.2 Mid-winter nutrient observations, $\hat{N}_{w,s}$ at each station, s. 25

6.1 Top three ranking inputs based on the number of runs at which an input scores one of the top three derivatives magnitudes. The numbers in brackets denote the number of runs at which each input scores the first, second and third highest derivative magnitude. 86

6.2 R^2 values from OLS fits at station 1115. 88

6.3 Estimates for B coefficients from OLS fit. 89

6.4 Estimates from the modified spectral method at the exploratory site. . . . 91

7.1 Percentage of input points ruled implausible at each station based on n simulator runs and $\sigma_\eta = m\sigma_e$. F denotes the percentage when derivatives are not included and T when they are. 116

7.2 Implausibility scores at $\hat{x_0}$. Bold type denotes the output with maximum implausibility score at each station. 125

7.3 Implausibility of input $\hat{X}_0$ corresponding to best matching set of output for each station from the fifty simulator runs. 126

Chapter 1

Introduction

"So what is your PhD about?" I had spent a week working at the Daily Mirror and, with the exception of the enthusiastic musings of the managing editor that having 'a doctor' on the team might enable the resurrection of a long-forgotten weekly column addressing the medical (predominantly sexual) problems of the readership, it was the first time somebody had taken an interest.

Unsure as I always am about how much people want to know when they ask the question, I tentatively began to explain: *"Well, think about climate change and how scientists are trying to understand it - by building gigantic computer models of the earth's climate and running them forward into the future to see what happens. I'm interested in how we can use those kinds of computer models to learn about real world phenomenon such as climate change."*

"But don't those models rely on some phenomenal assumptions?" he asked.

For the little that many people understand about the intricacies of science, it is often surprising how adept they can be at pointing out its weaknesses. The great minds of science toil to reproduce the complexities of the earth's climate on a computer; attempting what, to the layman, is obvious as being an impossible task.

The statistician, on the other hand, is lucky in being able to keep one step ahead of both the scientist and the layman. A good computer model of any complexity involves a combination of scientific expertise and understanding of some aspects of the physical process being modelled, together with some fairly rough and ready approximations about aspects which are not well understood and then, finally, a

jarring together of these aspects. The statistician avoids the jarring by recognising - and embracing - the uncertainties in such problems. Fairly recently, a whole literature has sprung up to address the statistical issues that arise from using computer models to learn about physical systems. The work allows various levels of information and scientific understanding on different aspects of physical phenomenon to be joined together in a coherent and rigorous manner. The glue that is used to do so is 'uncertainty'.

The Bayesian subjectivist view of probability interprets this uncertainty quite naturally in terms of the beliefs of the model builder who, although in some sense an *expert*, has various degrees of uncertainty about different aspects of the physical process being modelled. Uncertainty about the exact form of equations that govern physical dynamics induces uncertainty on the model output or 'prediction'. In other words, those 'phenomenal assumptions' are acknowledged, uncertainty purporting to them stated and the transfer of this uncertainty onto predictions of the future then tracked. Bayesian statistics provides a framework which allows an expert's beliefs and inherent uncertainty to be cobbled together in a cogent manner - and then updated as information becomes available. This is the starting point for our work.

The chronology of thought is not captured altogether in the thesis and it is perhaps worth explaining at this point. The statistical work on computer models treats the model as a 'black box', sending different choices of input - often corresponding to obscure unphysical quantities - into the model, enabling a probabilistic picture to be built up of how changing inputs changes output. A desire to 'open up' the black box is what ultimately led to the idea of inclusion of derivatives of simulators as extra information which we could use to update beliefs and reduce uncertainty. Opening up the black box meant looking at the equations and the numerical solution method employed to solve them. Doing this, it becomes clear that the quantities that are treated by the numerical solution method as numbers can more naturally be thought of as functions of inputs. Then, for example, Taylor expansions of these functions could be carried through the solution method in much the same way as the numbers. The coefficients of a Taylor expansion of a function are, of course, the

derivatives of the function.

Hence consideration of derivatives forms the basis of the original work in this thesis. From how to generate derivatives, how to estimate how much additional cost there will be in doing so, and how much extra information we can expect to gain by doing so. The aim is to demonstrate that anyone faced with a compartmental model of any complexity stands to benefit a great deal by performing an analysis of the type developed in this thesis.

A compartmental model of plankton cycles is used throughout to illustrate and motivate the work. The model was chosen for two reasons. Firstly, the model is an example of a wide class of *compartmental* computer models whose general features we are interested in exploiting through development of the statistical methodology. Compartmental models are popular and widely applied to model systems, such as the one under consideration, for which fundamental equations, describing the underlying physical processes, are yet to be established. Such an approach generally leads to a system of equations containing a large number of poorly known parameters. Calibrating these parameters to physical data is far from easy, particularly when observations at several locations are required to be matched (Hemmings et al., 2003, 2004). Secondly, building a generic model which captures the main features of annual plankton cycles at any location in the world ocean is a genuine physical problem that has long occupied those in the field of oceanography. The problem has received renewed attention recently in the context of climate change in which it is believed there is potential for significant climatic effects as a result of behaviour of the marine biota. Thus we chose the model in the belief that it would benefit from being set into a rigorous statistical framework such as the one we develop and, although our aim was not to solve the full problem in its entirety, we hoped to be able to offer some new insight of interest to those concerned with this specific physical problem.

The structure of the thesis is thus as follows. In Chapter 2, we describe in detail the computer model and calibration problem which is used to demonstrate the methodology and which serves as a motivation for the work.

Chapter 3 is a presentation of recent literature on Bayesian analysis of computer experiments which forms the theoretical starting point of our work. We detail the

Bayesian interpretation of computer models which views any such model as an unknown quantity about which we have beliefs, and in particular spend some time introducing 'Bayes Linear methods' as a natural way to describe these beliefs.

In Chapter 4, we demonstrate a natural extension of the approach described in Chapter 3 which allows us easily to incorporate information from derivatives of output with respect to the model input parameters. The extension is a completely general one which allows us to include derivative information for a process whenever it is available. Having established this, we go on to show two things. Firstly that derivatives offer us 'more of the same', in terms of providing extra information which can be used to reduce uncertainty. We perform some theoretical calculations to get an idea of how much additional information derivatives can be expected to tell us in terms of this reduction of uncertainty. Secondly, we show that this extra information is of a slightly different type to that which is usually available which enables us to target information at covariance parameter estimation; an aspect of the current statistical methodology which up until now has proved to be problematic.

In Chapter 5 we develop an efficient way to obtain derivatives for compartmental models which exploits the compartmental structure of such models. We also develop a heuristic for estimating the additional cost, in computing time, in generating derivative information for compartmental models, which we then test and discuss in light of the theoretical calculations of uncertainty reduction in Chapter 4.

Chapter 6 is where we apply our adapted methodology to the physical problem. The aims of the chapter are dual. Firstly, we are interested in the impact of derivative information on the process of building and refining of beliefs and we carefully compare situations with and without derivatives at various stages in order to understand the role of derivatives through this process. Secondly, we are interested in building up a picture of the simulator as best we can in order to produce some useful insights into the physical problem, of interest to those concerned with the model itself. Hence we discuss several problem-specific issues and present some solutions to these issues.

In Chapter 7, we calibrate the plankton model to physical data. Once again, our focus is two-fold. We are interested firstly in comparing analyses with and without

derivatives to assess the impact that derivatives have in the calibration process and, secondly, in the results of the calibration in relation to the physical problem. For the latter, we discuss the results in the light of previous analyses and show that they offer some interesting new insights.

Finally, in Chapter 8, we conclude with a discussion of our results and the most promising areas for future work. Code to perform updating and implausibility calculations throughout the thesis was developed in R (R Development Core Team, 2004) and is given in the Appendices.

Chapter 2

Calibrating the plankton cycle

The aims of this chapter are threefold. Firstly, we introduce and discuss a real world model calibration problem which serves as a motivation for the work in the remainder of this thesis and gives context for the recent statistical theory, outlined in Chapter 3, into which the problem falls. Secondly, we point out the features of the physical problem and model which we wish to exploit to build on current methodology for this class of problems. Thirdly, we give exact details of the model equations and data. Some of the model details, such as the model equations, may be skipped over by the reader without any loss of comprehension of subsequent parts of the thesis and, wherever this is the case, it is stated.

2.1 Motivating problem

Building a generic model which captures the main features of annual plankton cycles at any location in the world ocean is a challenge that has long occupied those in the fields of oceanography and marine biology. The problem has received renewed attention recently in the context of climate change in which it is believed there is potential for significant climatic effects as a result of behaviour of the marine biota. Typically the models considered are compartmental, in which each compartment represents an aggregated group of species assumed to be homogeneous. Compartmental models are popular and widely applied to model systems, such as the one under consideration, for which fundamental equations, describing the underlying physical processes,

are yet to be established. Such an approach generally leads to a system of equations containing a large number of poorly known parameters. Calibrating these parameters to physical data is far from easy, particularly when observations at several locations are required to be matched (Hemmings et al., 2003, 2004).

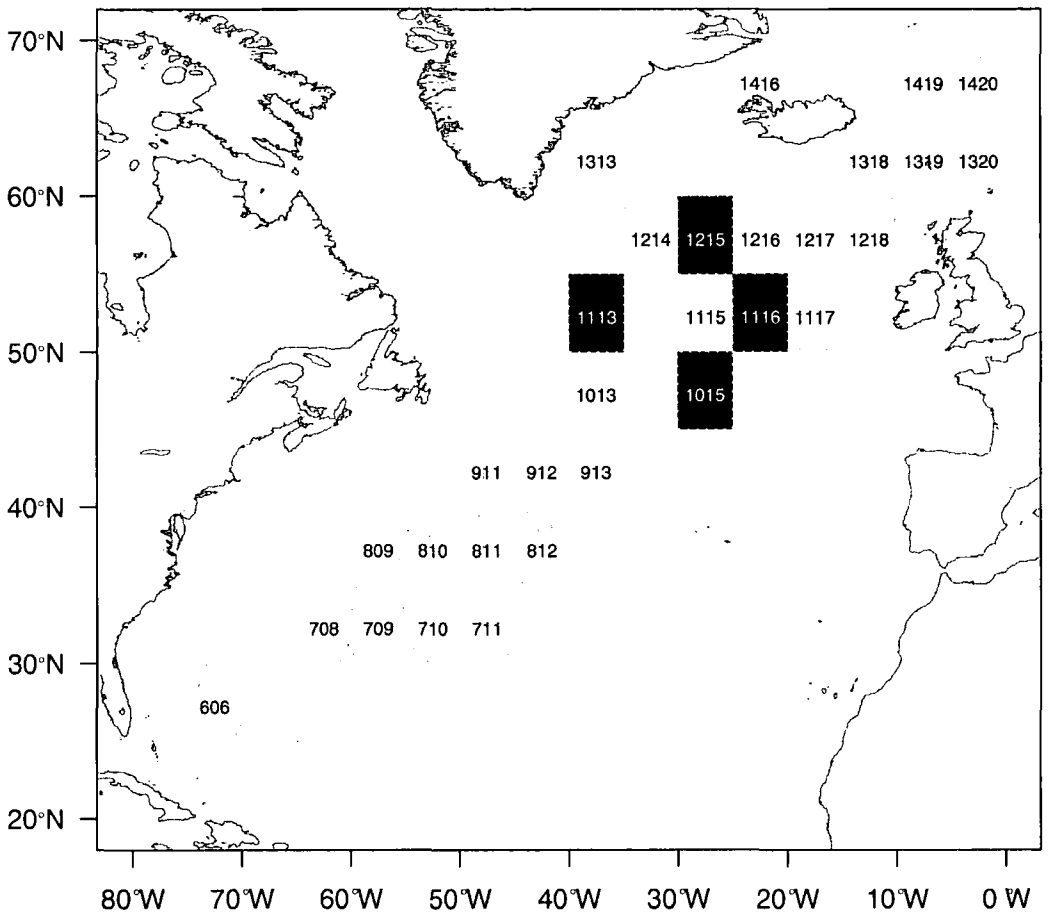


Figure 2.1: Map showing station locations, taken from Hemmings et al. (2004). The subset of stations marked in black is used throughout this chapter to illustrate a range of physical observations and location-specific simulator data.

Figure 2.1 shows the locations of the thirty stations considered in Hemmings et al. (2004). The station numbering is consistent with Hemmings et al. (2004),

who forms the numbers by dividing the map into a grid and concatenating the two digit meridional and zonal position numbers in the grid. The four stations marked black are used as 'demonstration' stations throughout this chapter to illustrate a range of forcing data and physical observations specific to each location.

Hence we have a physical system $y(\phi, t)$ which depends on a spatial component, ϕ, and a temporal component, t. As a representation of y, we have a computer model or *simulator* $s_l(x) = s(x, F_{l,t})$, where l indexes the spatial location ϕ_l, which is a function of an input vector x and a set of forcing functions $F_{l,t}$ where the simulator output, $s_l(x)$, is in general a vector of outputs corresponding to different time points. In our case, the spatial element $\phi = (\phi_x, \phi_y)$ is two dimensional with ϕ_x denoting longitude and ϕ_y denoting latitude. We consider a discrete number of locations, which we refer to as *stations*, and look for the simulator to reproduce the average variation in a given area about each station. In general, the act of spatial-averaging could induce a covariance structure across stations, although the resolution of forcing functions and physical data was considered to be high enough to ignore this effect here.

Our interest is in calibrating x to physical data across a collection of stations under the assumption that the forcing functions resolve the variation at different locations and that x, which governs plankton dynamics, is spatially and temporally invariant. The problem is a real-world physical problem taken from a series of papers (Hemmings, 2000; Hemmings et al., 2003, 2004). Hemmings provided run code from Hemmings (2000) and the physical and forcing function data used in Hemmings et al. (2003, 2004). On several occasions, his advice was sought as an 'expert' (in the Bayesian sense) and, as such, he is referred to as 'the expert' in the thesis wherever appropriate. As a disclaimer to this, it should be pointed out that the arrangement was only ever an informal one and any attributed specifications should be viewed with this in mind.

2.2 The model

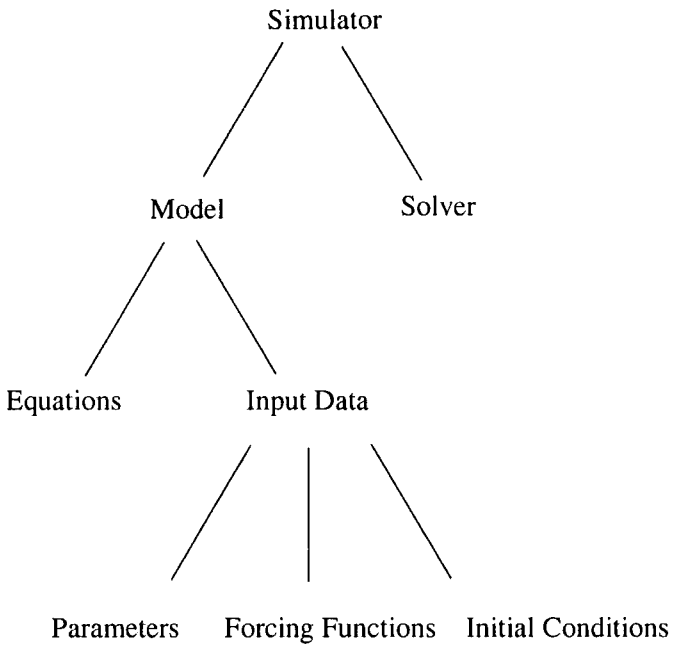
The model is a simple compartmental ecosystem model, taken from Hemmings (2000), which describes the evolution over time of three interacting populations

- phytoplankton, P
- zooplankton, Z
- nutrient, N

The concentration of each (in mmol Nm^{-3}) in the top 'layer' of the ocean, known as the *mixed layer* (see Section 2.5.1), is modelled as a homogeneous compartment. The model is essentially one of nitrogen flow, each population representing nitrogen molecules in a different form. The model equations, set out in Section 2.3, cannot be solved in closed form and a numerical solver or *computer simulator* routine is required to obtain an approximate solution. Given suitable input data, the computer simulator solves the system of equations to produce time series output for the three populations over a period of one year. The computer simulator is deterministic so that, if we were to run the solver routine at the same collection of input data on the same computer, we would obtain identical output. A single run of the simulator, for a given choice of input data, typically takes a few seconds.

Simulator input can be divided into three broad types. First there are the unknown input parameters, listed in Table 2.1, which appear on the functional relationships which govern inter-compartmental flows, along with imports and exports from the system. Second, spatio-temporal forcing data must also be specified before the model can run. These data — described in detail in Section 2.5 — are taken to be known and comprise time series, appropriate to the physical location being considered, derived from a variety of sources: day length, τ, is given by established theory, photo-synthetically available radiation, $I_0(t_D, t)$, from a combination of theory and physical observations of cloud cover and, finally, mixed layer depth, M, is given by empirical results of a general circulation model. The model assumes that there is no feedback from the system state to the forcing functions so that forcing function values are identical for different choices of the input parameters. This is

Figure 2.2: Schematic overview of the simulator.



justifiable for our model since any such feedback can be considered to take place over a much longer time scale than that considered for the purposes of this thesis. Finally, the simulator requires initial conditions to be specified for each population. In practice, these are unknown; however, here we treat them as fixed and run the simulator for a "spin up" period of one year after which time we consider the output to be effectively independent of these values. We take the fixed starting values $P(-365) = 0.02$, $Z(-365) = 0.002$, $N(-365) = 1$, used in Hemmings et al. (2004). Figure 2.2 gives an schematic overview of the different aspects of the simulator.

Output at the four demonstration locations for the default input settings are shown in Figure 2.3. A noticeable feature of the output, apparent in Figure 2.3, is the 'spikiness' of the phytoplankton and zooplankton time series. The spikes, which correspond to sudden blooms of the populations, are a well-established phenomenon of plankton dynamics. For each station shown in Figure 2.3, blooms of the phytoplankton prey are always followed by blooms of the zooplankton predator. By contrast, the output for nutrient, N, has no spikes although it registers a sharp dip in the middle of the year.

	Description	Units	L	D	U
ϕ_P	phytoplankton mortality rate	d^{-1}	0	0.05	0.3
k_N	nutrient uptake half-saturation constant	mmol N m^{-3}	0.05	0.5	1
V_P	phytoplankton maximum growth rate	d^{-1}	0	1.5	4
α	initial slope of P-I curve	$(lyd^{-1})^{-1}$ d^{-1}	0	0.05	0.2
k_P	light attenuation coefficient for phytoplankton	m^2(mmol N)$^{-1}$	0	0.03	0.3
β	zooplankton assimilation efficiency	-	0	0.75	1
μ	zooplankton excretion rate	d^{-1}	0	0.1	0.5
ϕ_Z	zooplankton mortality parameter	(mmol N m^{-3} d)$^{-1}$	0	0.2	0.3
g	zooplankton maximum ingestion rate	d^{-1}	0	1	3
k_G	zooplankton ingestion half-saturation constant	mmol N m^{-3}	0.05	1	3
ϵ	export fraction of zooplankton faeces	-	0	0.33	1
N_{ref}	sub-surface nutrient at reference depth	mmol N m^{-3}	3	9	15
m	cross-pycnocline mixing rate	m d^{-1}	0	0.2	1

Table 2.1: Inputs with Expert's Lower Bound, L, Default Value, D, and Upper Bound, U.

Figure 2.4 shows the model structure; in crude terms, phytoplankton feed on nutrient, zooplankton feed on phytoplankton and zooplankton faeces flow back as nitrogen into the nutrient pool. Exports from the system in the form of dead plankton and faeces from the phytoplankton flow from both P and Z to N and are then immediately exported from the system. The rate at which these processes take place is affected by the amount of sunlight, which affects the rate of phytoplankton growth, and the depth of the mixed layer, which kills both phytoplankton and zooplankton as it decreases.

The arcs in Figure 2.4 correspond to 'flow functions' which determine the rate of flow to and from compartments. Each arc represents an additive part of the flow so that Figure 2.4 specifies the following system of equations which describe the

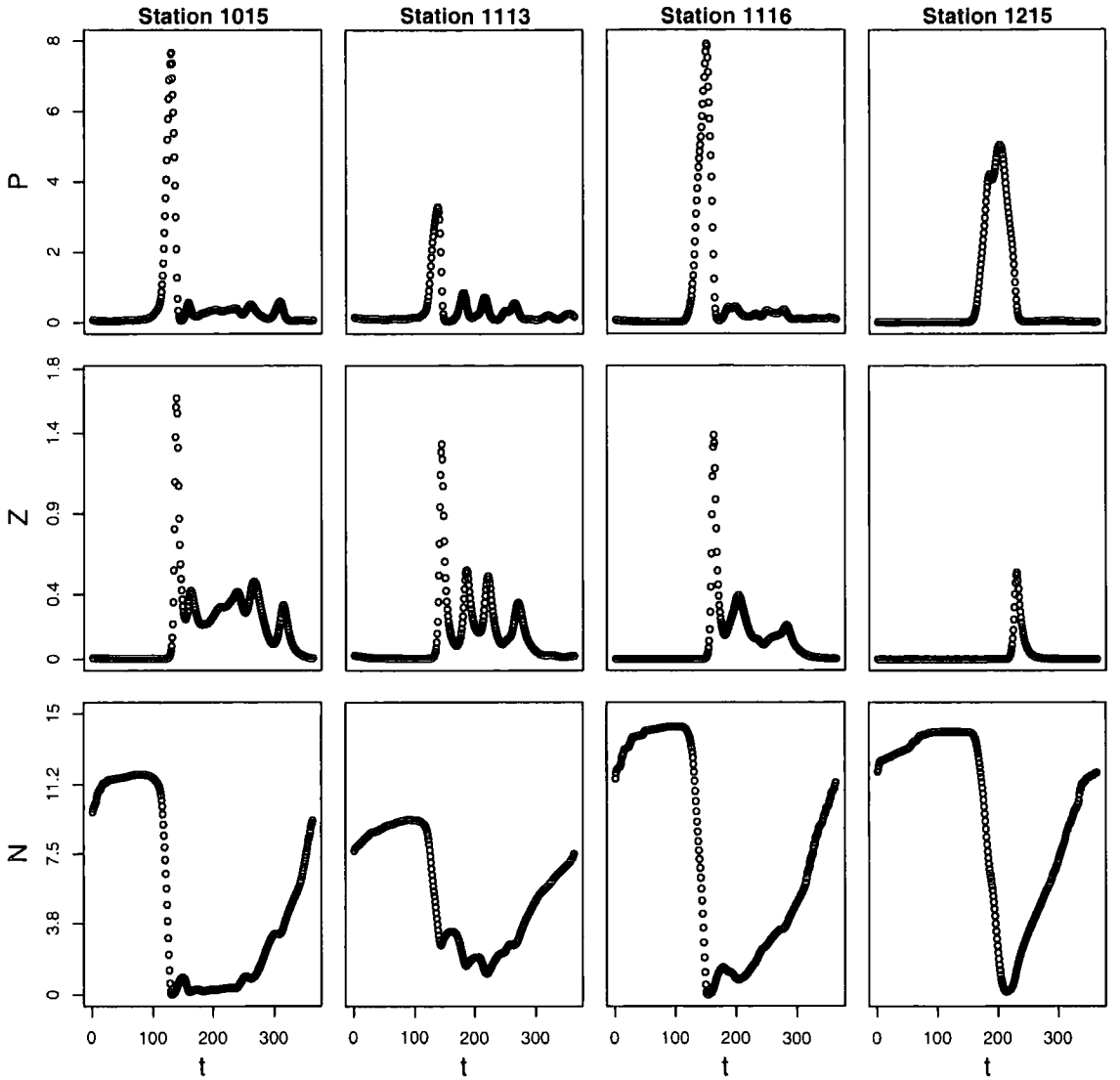


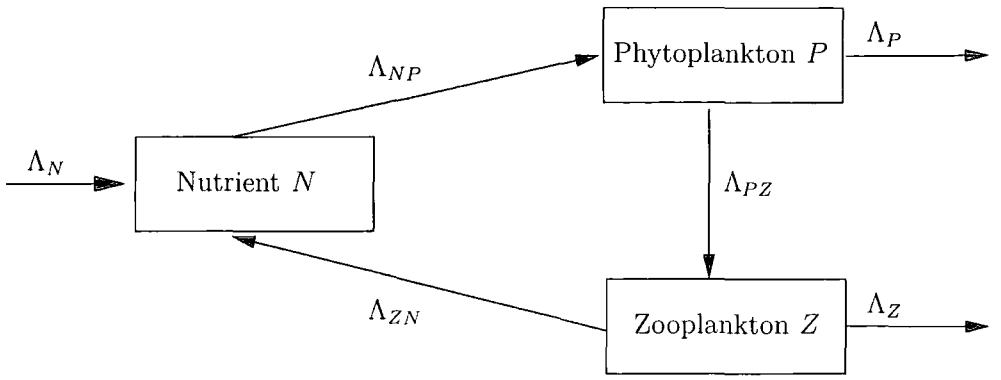
Figure 2.3: Simulator output for the default input setting plotted against time (days) at the four demonstration stations. The first row corresponds to phytoplankton, $P(t)$, the second row to zooplankton, $Z(t)$, and the third row to nutrient, $N(t)$.

variation in P, Z and N over time, t.

$$\begin{aligned} \frac{dP}{dt} &= \Lambda_{NP} - \Lambda_{PZ} - \Lambda_P \\ \frac{dZ}{dt} &= \Lambda_{PZ} - \Lambda_{ZN} - \Lambda_Z \\ \frac{dN}{dt} &= -\Lambda_{NP} + \Lambda_{ZN} + \Lambda_N \end{aligned}$$

The flow functions depend on P, Z and N and on the input data and, whilst the dependencies can be complex, they are specified by the model equations in

Figure 2.4: Schematic diagram of the simulator.



closed form. The model equations describing these explicit dependencies are given in Section 2.3. In general, we are interested in models of this type and the reader can skip through Section 2.3 without impeding their understanding of subsequent sections of the thesis.

2.3 Equations for the PZN simulator

The time-dependent relationships between the three populations, P, Z and N are described formally by the following state equations. All quantities which do not appear as inputs in Table 2.1 are time dependent unless stated otherwise.

2.3.1 Phytoplankton

The rate of change of phytoplankton, P, is given by the equation

$$\frac{dP}{dt} = PQ\bar{J} - G_P - \phi_P P - \frac{(m + h^+)P}{M} \tag{2.1}$$

where

- The first term is primary nutrient production, corresponding to Λ_{NP} in Figure 2.4, and is a product of three factors:

  (i) Phytoplankton concentration, P.

  (ii) The nutrient limitation factor,

$$Q = Q(N) = \frac{N}{k_N + N}, \tag{2.2}$$

parameterised by the *half saturation constant* for nutrient, k_N; the concentration at which the growth rate of the nutrient population is half its maximum rate.

(iii) The light-limited growth rate, $\bar{J}$, determined by

$$\bar{J}(t,P) = \frac{1}{M}\int_0^\tau \int_0^M J(z,t,t_D,P)dzdt_D \tag{2.3}$$

where t_D is the time of day,

$$J(z,t,t_D,P) = \frac{V_P \alpha I(z,t,t_D,P)}{\sqrt{V_P^2 + \alpha^2 I(z,t,t_D,P)^2}} \tag{2.4}$$

is the light-limited growth rate at depth, z, and time, t, and

$$I(z,t,t_D,P) = I_0(t,t_D)\exp\{-(k_W + k_P P)z\} \tag{2.5}$$

is the underwater light field where $k_W = 0.04$ is the light attenuation coefficient for water, which we treat as known, and k_P is the (unknown) light attenuation coefficient for phytoplankton. $I_0(t,t_D)$ is a forcing function described in Section 2.5.2.

- The second term in (2.1), G_P, is the grazing rate of zooplankton on phytoplankton, corresponding to Λ_{PZ} in Figure 2.4 which is given by

$$G_P = G_P(P,Z) = \frac{gZP}{k_G + P} \tag{2.6}$$

and parameterised by the zooplankton maximum ingestion rate, g, and the half saturation constant for zooplankton ingestion, k_G.

- The final two terms in (2.1) are exports from the system, the sum of which corresponds to Λ_P in Figure 2.4. The first term models phytoplankton mortality and is parameterised by the phytoplankton mortality rate ϕ_P. The second export term models the physical flux due to vertical mixing processes, and depends on the the rate of diffusive mixing across the mixed layer base, m, the mixed layer depth forcing function, M, and $h^+ = \max(h,0)$, where $h = \frac{dM}{dt}$.

2.3.2 Zooplankton

The rate of change of zooplankton, Z, is given by

$$\frac{dZ}{dt} = \beta G_P - \mu Z - \phi_Z Z^2 - \frac{(m + h^+)Z}{M} \tag{2.7}$$

where

- G_P is the grazing rate of zooplankton on phytoplankton given in (2.6) and β is the assimilation efficiency from phytoplankton to zooplankton.
- The remaining three terms form the zooplankton exports (Λ_Z in Figure 2.4), the first being parameterised by μ, the zooplankton excretion rate; the second by ϕ_Z, the zooplankton mortality parameter; and the third representing the physical flux which takes the same form as that for phytoplankton in (2.3.1).

2.3.3 Nutrient

The rate of change of nutrient, N, is given by

$$\frac{dN}{dt} = -PQ\bar{J} + \mu Z + (1 - \epsilon)(1 - \beta)G_P + \frac{m + h^+}{M} \max[N_{\text{sub}}(M) - N, 0] \tag{2.8}$$

where

- ϵ is the exported fraction of zooplankton faecal material.
- $N_{\text{sub}}(z) = N_{\text{ref}} \ln(bz + 1)$ is the subsurface nutrient concentration at depth z, parameterised by N_{ref}, the nutrient concentration at reference depth $(e-1)/b$, where we fix this reference depth at 100m by taking $b = 0.017$. This term corresponds to Λ_N in Figure 2.4.

2.4 Simulator input: invariant input parameters

Table 2.1 lists the set of invariant input parameters (referred to hereafter as 'inputs') collected in the model equations in Section 2.3. The value of each of the inputs is uncertain and this is reflected by the specification of lower and upper bounds for each by our expert, listed in Table 2.1. Zero lower bounds are strict because of the

requirement that parameters be non-negative whilst other bounds arise from a combination of previous informal data analysis experiments and knowledge of observable quantities which are considered to correlated with the inputs. Positive values outside of these ranges are considered by the expert to be "unphysical" although, because of deficiencies in the model and forcing data, relating the inputs to real-world quantities is ambiguous and so values outside the ranges which produce realistic output are also of interest.

Note that, in general, it is desirable to extract as much information as the expert is confident to give about input values; for example, a 95% percentile would have been desirable had the expert been willing to specify one. The issue is that whilst in some sense it is desirable for the expert to give conservative bounds for input values, the downside to this is that "unphysical" values of inputs may have high leverage in fitting the emulator because of their position on the extremities of the input space; in other words, we want to build as good as fit as possible on the range of interest rather than compromising to include values in which, ultimately, we are not interested. In addition, information about whether any depedencies exist between inputs is desirable but again generally (as in our case) difficult to elicit.

2.5 Simulator input: physical forcing

The model is driven by *forcing functions*, imposed at each location $\phi = (\phi_x, \phi_y)$, and determined by external data in the form of time series of

- Mixed Layer Depth (MLD), $M[\phi_x, \phi_y, t]$.
- Photosynthetically Available Radiation directly below the sea-surface (PAR), $I_0[\phi_x, \phi_y, t]$.
- Day length, $\tau(\phi_y, t)$.

Each of the three quantities is in general continuously dependent on location, ϕ, and time, t. Our notation is used to to demonstrate whether the data available to us is discrete (MLD and PAR, shown with square brackets) or in continuous closed form (day length, curly brackets).

The forcing values are taken to be known throughout this thesis, although this is clearly a simplification and, in general, forcing functions can themselves contain large amounts of uncertainty.

2.5.1 Mixed Layer Depth

The model describes plankton evolution in the *Mixed Layer*; the volume of water directly below the surface of the ocean outside of which it is assumed (by the model) that plankton is unable to survive. Formally, the *Mixed Layer Depth (MLD)* is defined in one of the following two ways:

- **Temperature Criterion** The depth at which the temperature falls to 0.5°C below that at the sea surface.
- **Density Criterion** The depth at which the density difference from the surface is 0.125 sigma units.

The mixed layer time series in our model are based on the temperature criterion and are taken from the output of a climatologically forced ocean general circulation model (Jia, 2000) corresponding to the final year of a 16 year integration of the coarse resolution (4/3°) version. Figure 2.5 plots the MLD series for the four demonstration locations. The minimum values of the MLD series at the four stations span the range 22m-30m and the maximum values 262m-550m. This compares to 20m-30m for minima and 225m-801m for maxima over the thirty stations in Figure 2.1. Comparing Figure 2.5 with the simulator output at the default input settings in Figure 2.3, we see that the stations with larger MLD levels tend to result in larger nutrient levels.

The MLD series, M, is also used by the simulator in forming $h^+ = \max(0,\, dM/dt)$ with the dM/dt series formed by differencing values of M.

2.5.2 Photosynthetically Available Radiation

The Photosynthetically Available Radiation (PAR) available directly below the sea-surface, $I_0(t, t_D)$, is in general a function of the time since the start of the year, t,

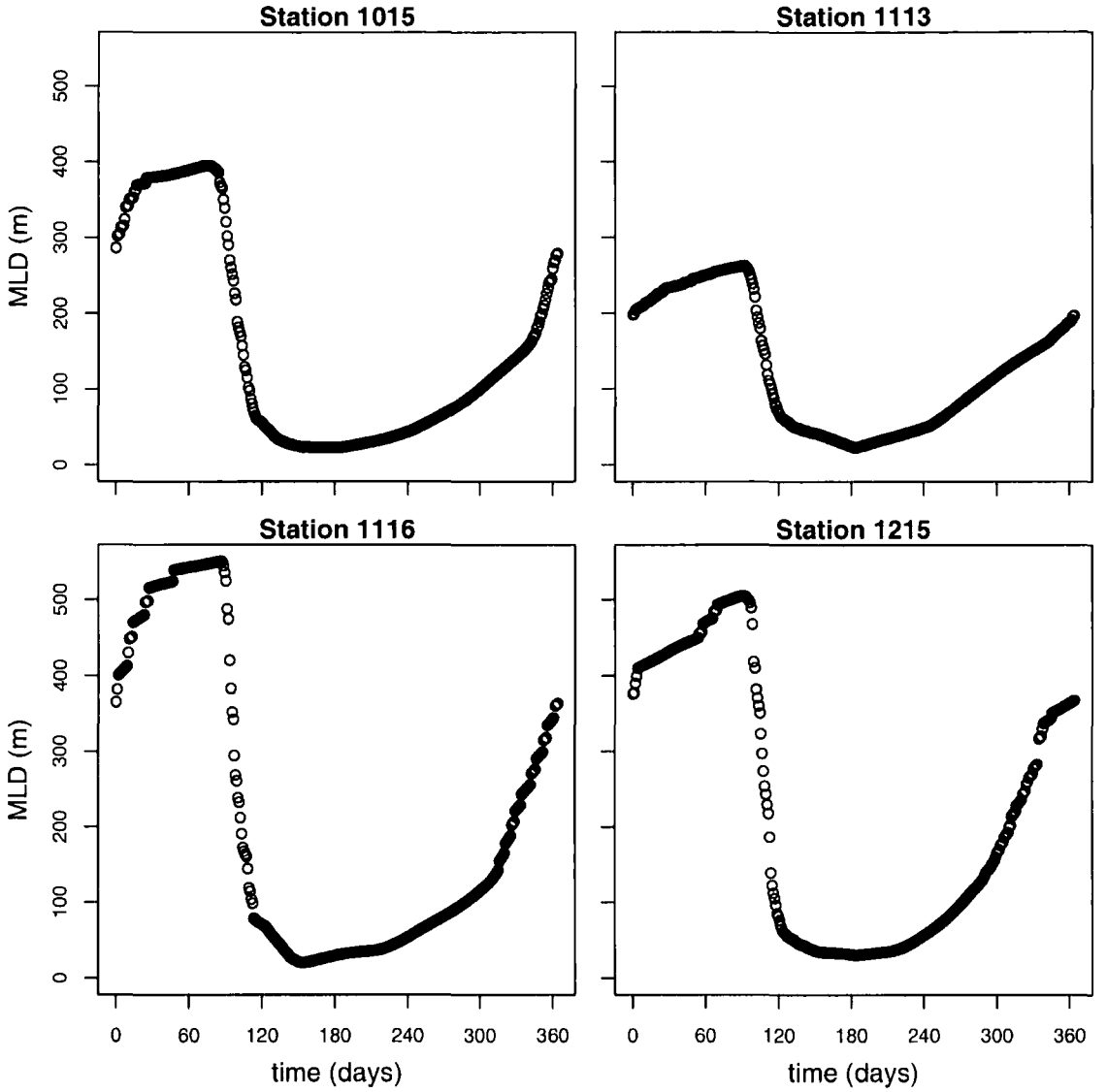


Figure 2.5: Mixed layer depth (m) against time (days) at the four stations.

and the time of day since sunrise, t_D. For the purposes of integrating, the following product form is taken

$$I_0(t, t_D) = I_0(t) f_D(t_D)$$

where

$$f_D(t_D) = \begin{cases} \frac{4}{\tau^2} t_D & \in [0, \frac{\tau}{2}] \\ \frac{4}{\tau^2}(\tau - t_D) & t_D \in [\frac{\tau}{2}, \tau] \end{cases}$$

PAR forcing for $I_0(t)$ was taken from SeaWiFS PAR Standard Mapped Image data of 8-day mean values for the year 1998 taken at a resolution of 9km. For each available time point, a value was computed for each station by averaging values

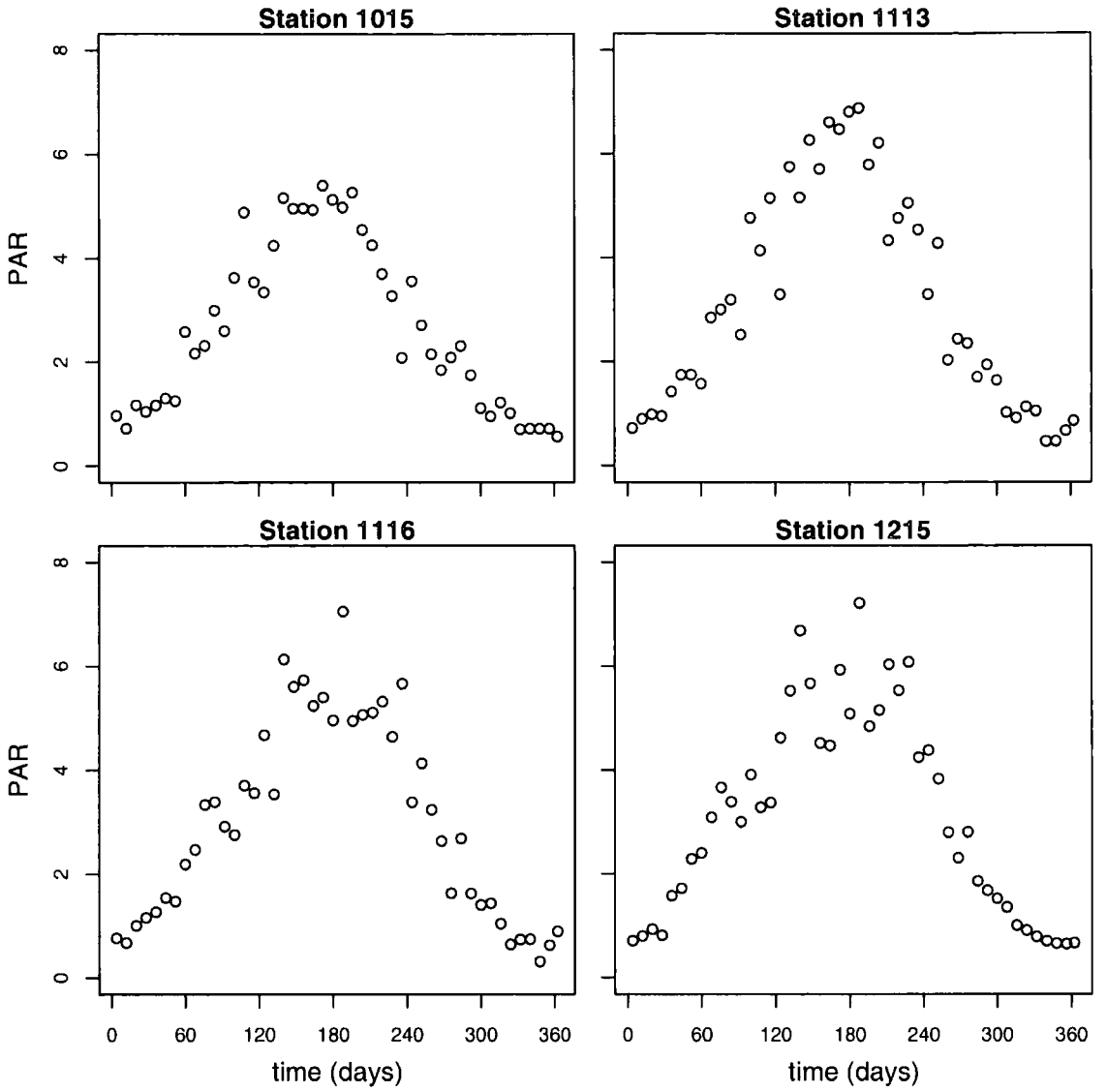


Figure 2.6: Photosynthetically Available Radiation against time (days) at the four stations.

over a 100km radius about the station. Series at the four demonstration stations are shown in Figure 2.6. We see that, at each station, PAR rises to a peak in mid summer before falling again.

2.5.3 Day length

Day length, τ, as a function of latitude, ϕ_y, and number of days since the start of the year, t, is given by (Brock, 1981)

$$\tau(\phi_y, t) = \frac{\arccos\left(-\tan\delta(t)\tan\phi_y\right)}{2\pi} \tag{2.1}$$

where

$$\delta(t) = 0.40928 \sin\left(2\pi\left(\frac{284+t}{365}\right)\right) \tag{2.2}$$

is the angle between the sun and equator, in radians, at solar noon. Note that 0.40928 rad = 23.45° is the angle between the earth's axis of rotation and the direction perpendicular to its orbit.

2.6 System observations

Past observations from the physical system at different locations are available which might be used to calibrate the unknown input parameters. The observations are those used in Hemmings et al. (2003) received in their final form after the processing described in this section had been carried out.

For each valid station, we have:

- A time series of chlorophyll a observations as an estimate of phytoplankton concentration at the corresponding time-points.
- Climatological nitrate and mixed layer depth data used to produce an estimate of the winter-time nutrient concentration; i.e. this gives us a single value to calibrate against.

2.6.1 Chlorophyll Data

The data consists of time series of chlorophyll a observations for the year 1998, derived from satellite observations made by SeaWiFS Standard Mapped Image ocean colour data at a resolution of 9km (McClain et al., 1998). The ratio of chlorophyll

to phytoplankton is taken to be 1 mol N per g chlorophyll so that the quantities correspond exactly.

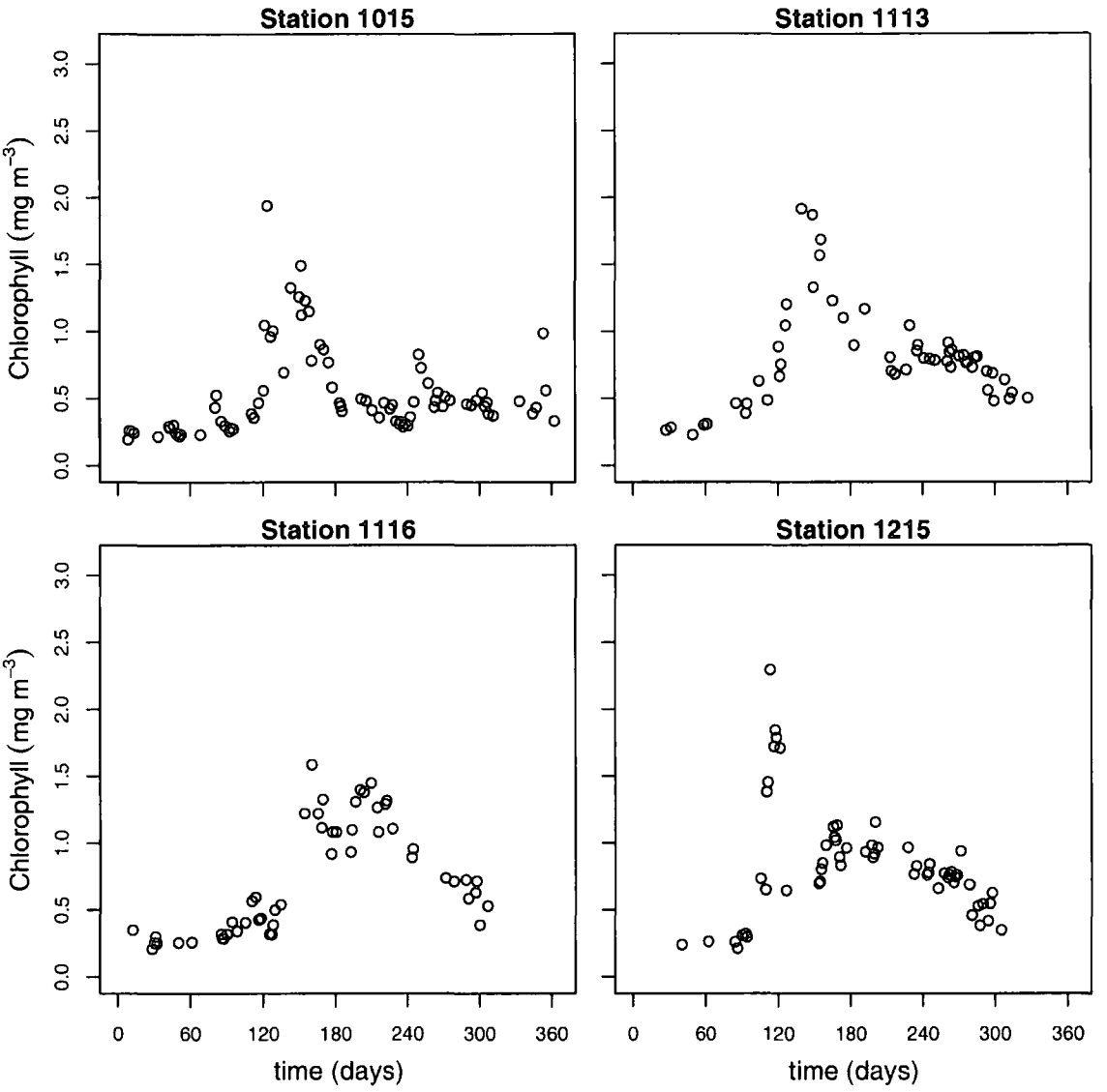


Figure 2.7: Observed chlorophyll (mmol m^{-3}) at the four locations recorded for 1998 where $t = 0$ corresponds to midnight GMT on 31st December 1997.

Time series for each of the thirty stations in Figure 2.1 were derived by averaging all valid pixels within a 100km radius of the corresponding station's position. Figure 2.7 shows the resulting series for the demonstration stations. In broad terms, we see a noticeable bloom in spring, with levels then either rapidly dropping again (as in Stations 1015 and 1215) or being maintained at elevated levels through the summer before a gradual decline in the autumn (as in stations 1113 and 1116).

The phytoplankton observation series contain various sources of measurement error and uncertainty which are difficult to quantify. In particular, the original chlorophyll series are the result of complex processing algorithms applied to the ocean colour data - itself, a combination of light wave signals from numerous sources - in order to extract the part of the signal corresponding to chlorophyll. Figure 2.8 illustrates the complexity of the task at hand, showing the main sources known to contribute to the signal detected by the satellite sensor. Light which is scattered into the sensor by molecules and aerosols in the intervening atmosphere can be up to as much as 95% of the total signal so that errors in "atmosphere correction" algorithms, which seek to remove this signal, can lead to significant errors in the processed data sets (Gordon, 1993). The target of SeaWiFS is to estimate the phytoplankton concentrations to within $\pm 35\%$ of their true values, where validation is carried out by comparison with more accurate readings taken from cruises and bouys. The attraction of ocean colour data, compared to the more accurate *in situ* measurements, is the extensive spatial and temporal coverage offered so, although less accurate, much more data is available to calibate against. We can expect to reduce these errors through our averaging of pixels, although by how much is unclear since values at adjacent pixels are unlikely to be independent.

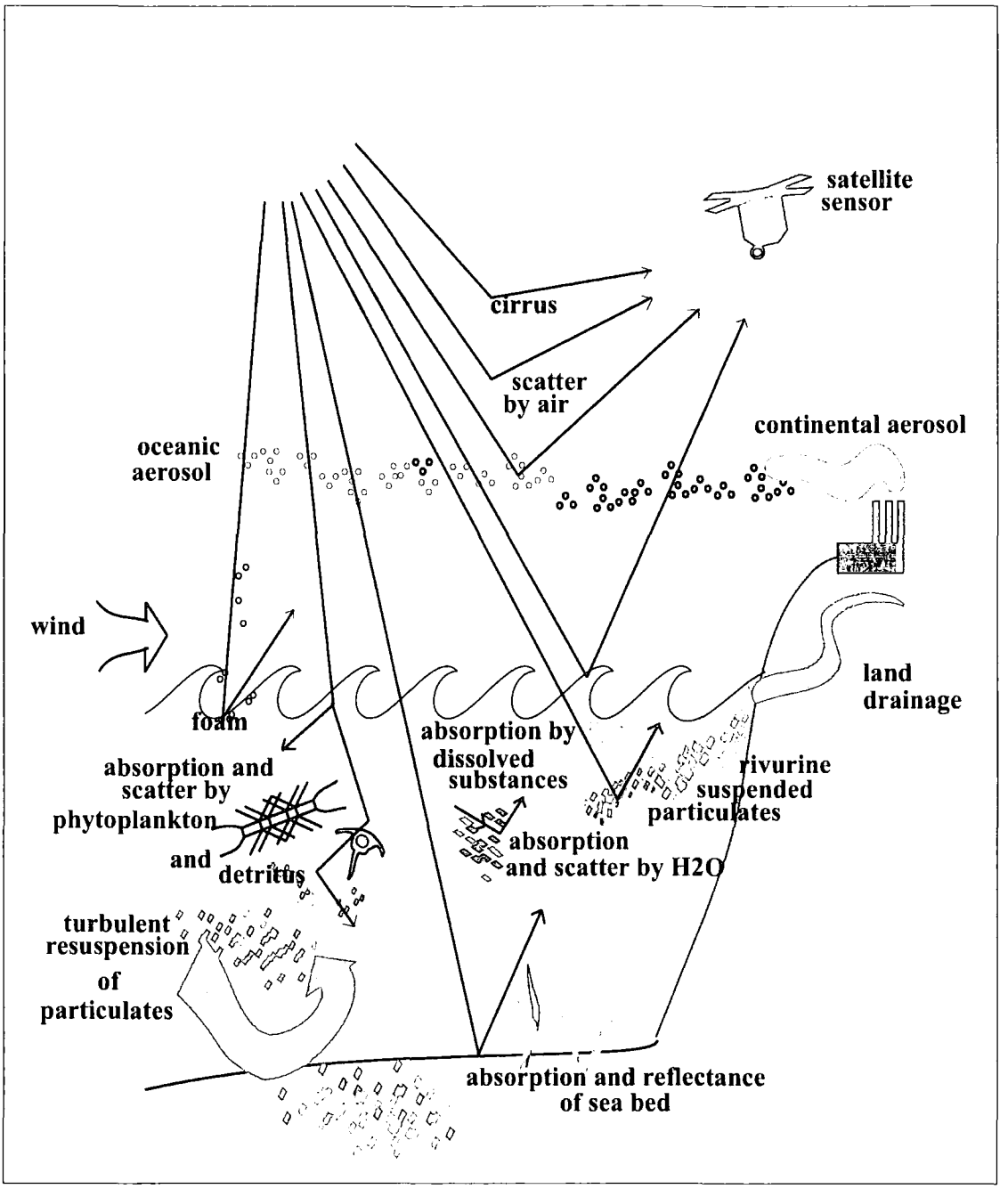


Figure 2.8: Optical interactions for Ocean Remote Sensing. Reproduced with permission from Andreas Neumann and Jill Schwarz.

2.6.2 Winter time Nutrient data

At twenty five of the stations, a single nutrient observation, $\hat{N}_{w,s}$, is obtained by treating a climatological estimate of the winter time nitrate level as an observation on the winter time nutrient level. In general, the nutrient compartment, which accounts for nitrogen in dissolved form, is a combination of nitrate and ammonium. In winter, however, the concentration of ammonium is assumed to be negligible and the nitrate value can be taken as an estimate of $N_{w,s}$, the nutrient value at the corresponding time point.

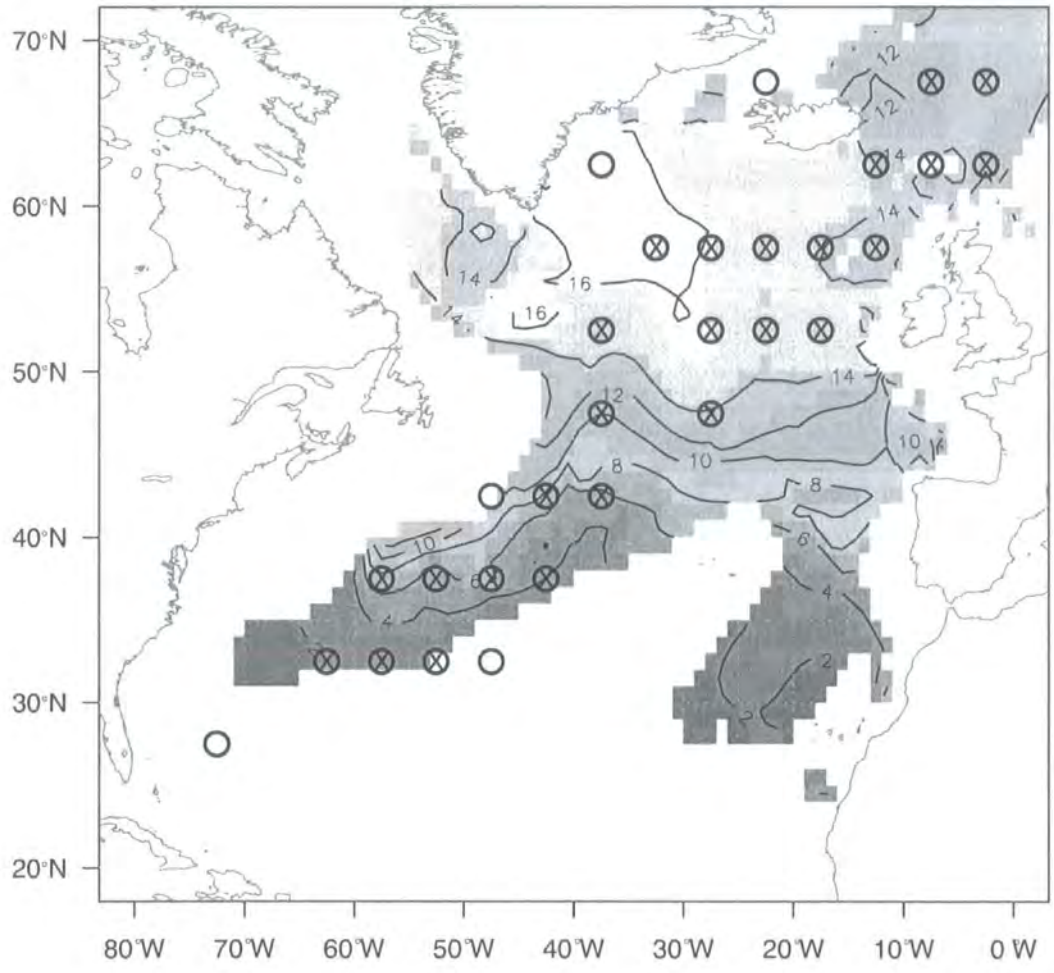


Figure 2.9: Estimated annual maximum nutrient concentration in the mixed layer (mmol m^{-3}). Circles denote stations and additional crosses mark stations where nitrate estimates are available.

s	606	708	709	710	711	809	810	811	812	911
$\hat{N}_{w,s}$	-	1.95	2.66	2.50	-	6.35	5.82	6.03	4.04	-
s	912	913	1013	1015	1113	1115	1116	1117	1214	1215
$\hat{N}_{w,s}$	8.42	6.16	10.63	13.77	15.12	15.85	14.68	15.23	16.5	16.03
s	1216	1217	1218	1313	1318	1319	1320	1416	1419	1420
$\hat{N}_{w,s}$	15.1	14.51	13.32	-	14.26	13.64	13.06	-	12.44	13.32

Table 2.2: Mid-winter nutrient observations, $\hat{N}_{w,s}$ at each station, s.

Estimated annual maximum nitrate values were obtained by averaging climatological estimates of mixed layer depth over the period February-April and then interpolating annual vertical nitrate profiles to this depth. The vertical nitrate profiles were taken from World Ocean Atlas 1° analysed annual mean fields (Conkright et al., 1998) and the mixed layer depth estimates from monthly data also on a 1° grid (Levitus, 1982). Locations with a winter-time mixed layer depth of less than 100m were discarded because nitrate concentrations above this depth are known to exhibit strong seasonal variation. Estimates for each station were obtained by averaging the nitrate field over a 3×3 grid centered on the corresponding station location.

Figure 2.9 shows the resulting annual nitrate maximum field with station locations laid over. Five of the thirty stations - 606, 711, 911, 1313, 1416 - were omitted because more than half the points in the 3×3 grid centered on these locations contained no observation. These omissions are denoted in Figure 2.9 by the absence of a cross inside the circle at these locations. The resulting values for stations are given in Table 2.2.

As with the chlorophyll observations, the $\hat{N}_{w,s}$ contain errors from various sources, including in the original mixed layer and vertical nitrate profile data and in the interpolation of this data. We expect to reduce the combination of errors through the averaging, although by how much will again be difficult to quantify.

2.7 Recap: General features of the problem

To conclude the chapter, we note that in general we are interested in models formed from a series of linked compartments, the form of whose equations are known but contain unknown input parameters which we wish to calibrate using physical data. Although given in closed form, the equations are coupled and non-linear so that their solution is not obtainable analytically but only approximately using a numerical solver. This last consideration is the norm for almost all compartmental models of any practical relevance. For an investigation of uncertainty in linear compartmental models which relies on exact, closed form solutions of outputs in terms of inputs being obtainable, see Cooke and Kraan (2000a,b).

Chapter 3

Computer Simulators and Emulation

The model and calibration problem outlined in Chapter 2 is essentially a statistical one. In this chapter we introduce and advocate a rigorous statistical approach to dealing with the uncertainties in problems of the type outlined in Chapter 2 and describe relatively recent statistical methodology for doing so. Our standpoint remains Bayesian throughout as we believe this offers the only natural interpretation of various aspects of the problem. The final two sections introduce the Bayes Linear methodology which, whilst Bayesian in spirit, drops full distributional assumptions required for beliefs in a full Bayes analysis and replaces them with low order belief structures which we believe to be more appropriate to problems of this type.

3.1 Computer Simulators

In general, a computer simulator, $s(\cdot)$, for a physical system, y, takes some input, x, and produces output, $s(x)$, as a representation of y. Access to such a simulator, together with historical observations from the system itself, gives rise to two immediate questions: "Which inputs produce outputs which closely match historical data?" (Calibration) and "How can we combine historical data and simulator output to predict future system behaviour?" (Forecasting). There is a considerable literature in this area; see Sacks et al. (1989) and Currin et al. (1991) for an early re-

view and Bayesian interpretation. Bayesian approaches are outlined and developed in detail in Craig et al. (1997) and Kennedy and O'Hagan (2000) whilst Santner et al. (2003) provides a recent and comprehensive overview of the field.

The need for such work is a result of the many difficulties associated with using such simulators for calibration and forecasting. Computer simulators are used to model complex physical systems. They offer the opportunity to perform experiments relating to systems for which direct physical experiments on the system are either impossible or prohibitively expensive. At the same, they also often encode much uncertainty from various sources, which we summarise below.

3.1.1 Uncertainty on inputs

Inputs to the simulator are unknown. Some inputs may relate directly to a physical quantity which is measurable (with error), but often - as with many of the parameters in our model, summarised in Table 2.1 - they will be poorly known and immeasurable, often relating to unphysical, aggregated quantities. This is a typical feature of compartmental models. Calibration aims to find combinations of input parameters that give rise to simulator output which closely matches historical observations. This can be thought of in terms of learning about the value of the 'true' input. The hypothesis that such a 'true' input exists (we adopt the term 'best input' for reasons which will become clear) raises several issues, primarily because the model is not an exact representation of reality, but rather there exists model discrepancy.

3.1.2 Model discrepancy

A computer simulator is a simplified representation of a physical system based on approximate science. For example, our model is a simplified form for which there are various more complicated models which themselves, although more accurate, are far from being exact (Hemmings et al., 2004). This means that, even if we were to remove all uncertainty relating to the inputs to the simulator, we would not expect the simulator to predict exactly the physical system. We will therefore

be interested in the 'discrepancy' between simulator and system, which we might define, for example, as the distance (in some metric) between the physical system and simulator at the best inputs.

The assumption of a best input is reasonable if the simulator is a good representation of the system and, from our point of view, it provides a device which allows us to link simulator and system together and proceed. The interested reader might consult Goldstein and Rougier (2004) for an in depth discussion of related issues.

3.1.3 Observation error

Observations from the physical system itself usually come with observation error. As is the case with the observations presented in Section 2.6, this uncertainty can be a combination of uncertainties from various sources and the error structure will often be complicated.

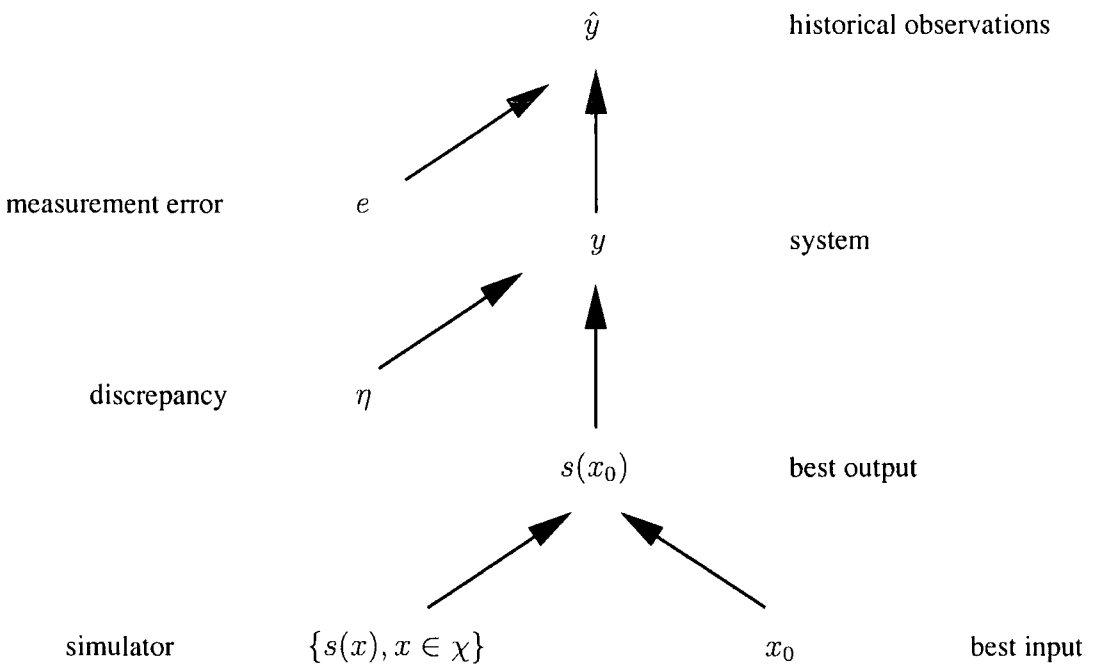


Figure 3.1: Conditional Independence Graph for the simulator, $s(\cdot)$, the best input, x_0, the physical system, y, and observations of the system, $\hat{y}$. Any two nodes are independent given the values of all their parent nodes.

Figure 3.1 shows a possible way to link the simulator to the system through these sources of error together using a Bayesian Graphical Model (see e.g. Cowell et al., 1999) and captures the essence of the problem: to use a simulator, $s(\cdot)$, and observations, $\hat{y}$, to learn about a physical system, y. The implications of the exact form of the Graphical Model are discussed in general and for our physical problem at the start of Chapter 7. A Bayesian interpretation, which we keep to throughout this thesis, is natural because of the subjective nature of the various aspects of the problem outlined and the consequent difficulty in describing them 'classically'. In particular, the discrepancy term, which represents our beliefs about the difference between simulator and system from a Bayesian perspective, is irreducibly subjective in the sense that there is no experiment that we can do to learn about it. In addition, the simulator we consider is deterministic so that running the code at the same set of inputs always produces the same output. However, as it appears in Figure 3.1, the implicit assumption is that $s(\cdot)$ is in some sense random. Once again, this apparent contradiction can be dealt with naturally using a Bayesian argument since we can consider the simulator as 'unknown to us'. In particular, since we can only observe the simulator at a finite number of input points, we are uncertain about the simulator output values over most of the space of inputs. The implications of this fact leads us to highlight a further area of uncertainty.

3.1.4 Simulator uncertainty

Simulators often contain a large number of poorly known input parameters and can also be expensive, in CPU time, to run. When either condition applies, we can only hope to get a thin coverage of the input space by evaluating the simulator at different input choices. Moreover, the simulators are often complex so that the effect that changing inputs has on simulator output is far from transparent. For the model described in Chapter 2, although the run time takes only a few seconds, there are thirteen unknown input parameters so that even a very sparse 3^{13} factorial design at a single station would require over a year of computing time (and we have thirty stations!) This leads onto uncertainty about the value of simulator output, which can be represented by building an *emulator*.

3.2 Emulators

In cases when the simulator is such that only a thin coverage of the input space is possible, a popular approach has been to construct a surrogate for the simulator, termed an *emulator*, as a stochastic representation of our beliefs about the deterministic simulator. Early work based on this approach is reviewed in Sacks et al. (1989). A common approach to emulators is to write down a representation of the output, $s(x)$, of the following form

$$s(x) = Bg(x) + \gamma(x) \tag{3.1}$$

In the presentation here for simplicity we consider $s(x)$ to be univariate although the extension to multivariate forms is, in principle, straightforward. The first term in (3.1) describes the main global effects of the inputs, in which $g(x)$ is a known vector of functions and B is a set of unknown coefficients. The second term, $\gamma(x)$, is a random process constituting a correlated error structure, which is taken to be uncorrelated with B. (In reality this is a simplifying assumption and B and γ are correlated since $s(x)$ is deterministic; it is justified by the fact that any contribution from associated terms is small when variation accounted for by γ is small and because, in any case, the dropped terms have little affect once we begin to make observations on $s(x)$.)

A very popular choice is to take $\gamma(x)$ to be a Gaussian process, which is characterised by the property that all finite dimensional probability densities are multivariate Gaussian

$$p_{\gamma_1,\ldots,\gamma_n}(x) = \frac{1}{(2\pi)^{\frac{n}{2}}(\det\Sigma)^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x - m(x))^T\Sigma^{-1}(x - m(x))\right) \tag{3.2}$$

and is thus determined completely by its mean function, $m(x) = \mathrm{E}[\gamma(x)]$, and covariance function, Σ, the $n \times n$ matrix with elements $\Sigma_{i,j} = \mathrm{Cov}[\gamma(x_i), \gamma(x_j)]$. We take $\mathrm{E}[\gamma(x)] = 0$ and $\Sigma = \sigma^2 R$ where σ^2 is a variance parameter and R is a $n \times n$ positive definite correlation matrix, $R_{i,j} = r(x_i, x_j)$. Typically, $r(\cdot,\cdot)$ is taken to be *isotropic* and *stationary* so that it is a function of the distance between points, $r(x_i, x_j) = r(||x_i - x_j||)$, where $r(0) = 1$ and $r(||x_i - x_j||)$ decreases monotonically to zero as $||x_i - x_j|| \to \infty$.

Popular choices for R are (for x scalar):

Power Exponential

$$r(x, x') = \exp\left(-\theta|x - x'|^p\right), \quad 0 < p \leq 2 \tag{3.3}$$

Matérn

$$r(x, x') = \frac{1}{\Gamma(\nu)2^{\nu-1}} \left(2\sqrt{\nu\theta}|x - x'|\right)^{\nu} K_{\nu}\left(2\sqrt{\nu\theta}|x - x'|\right) \tag{3.4}$$

where $K_{\nu}(x)$ is the modified Bessel function of the second kind, order ν (see Appendix F for more details).

The parameter, θ, which appears in the Power Exponential and Matérn forms, is often referred to as a 'smoothness' parameter although, as is clear from the equations, it is in fact a scale parameter. The parameter, p, appearing in the Power Exponential form, is referred to as the 'shape' parameter, because of its effect on the shape of the correlation function. The parameter plays an important role in determining smoothness of solutions, since $p = 2$ yields infinitely differentiable realisations (*in the mean-square* - see Section 4.1 for more details) whilst $p < 2$ corresponds to realisations which are non-differentiable. In this thesis, our interest shall be in differentiable processes only and we restrict attention to the case $p = 2$. For the Matérn family, ν might also be more accurately described as a smoothness parameter than θ, determining the differentiability of the process: realisations are almost surely continuously differentiable with order $([\nu] - 1)$ where $[\nu]$ is the smallest integer that is greater than or equal to ν (sometimes referred to as the integer ceiling of ν).

A popular and widely-used special case of *both* families - corresponding to taking $p = 2$ in the Power Exponential class, and taking the limit as $\nu \to \infty$ in the Matérn class (see Appendix F) - is the *Gaussian correlation function* (given for multivariate x)

$$R(x, x') = \exp^{-(x-x')^T \Theta (x-x')}, \quad \Theta = \text{diag}(\theta_1, \ldots, \theta_p) \tag{3.5}$$

Figure 3.2 shows realisations from the Matérn and Gaussian families for varying parameter values. From the Figure, we see why θ is often - misleadingly, from our point of view - referred to as a smoothness parameter. We see that, as we increase

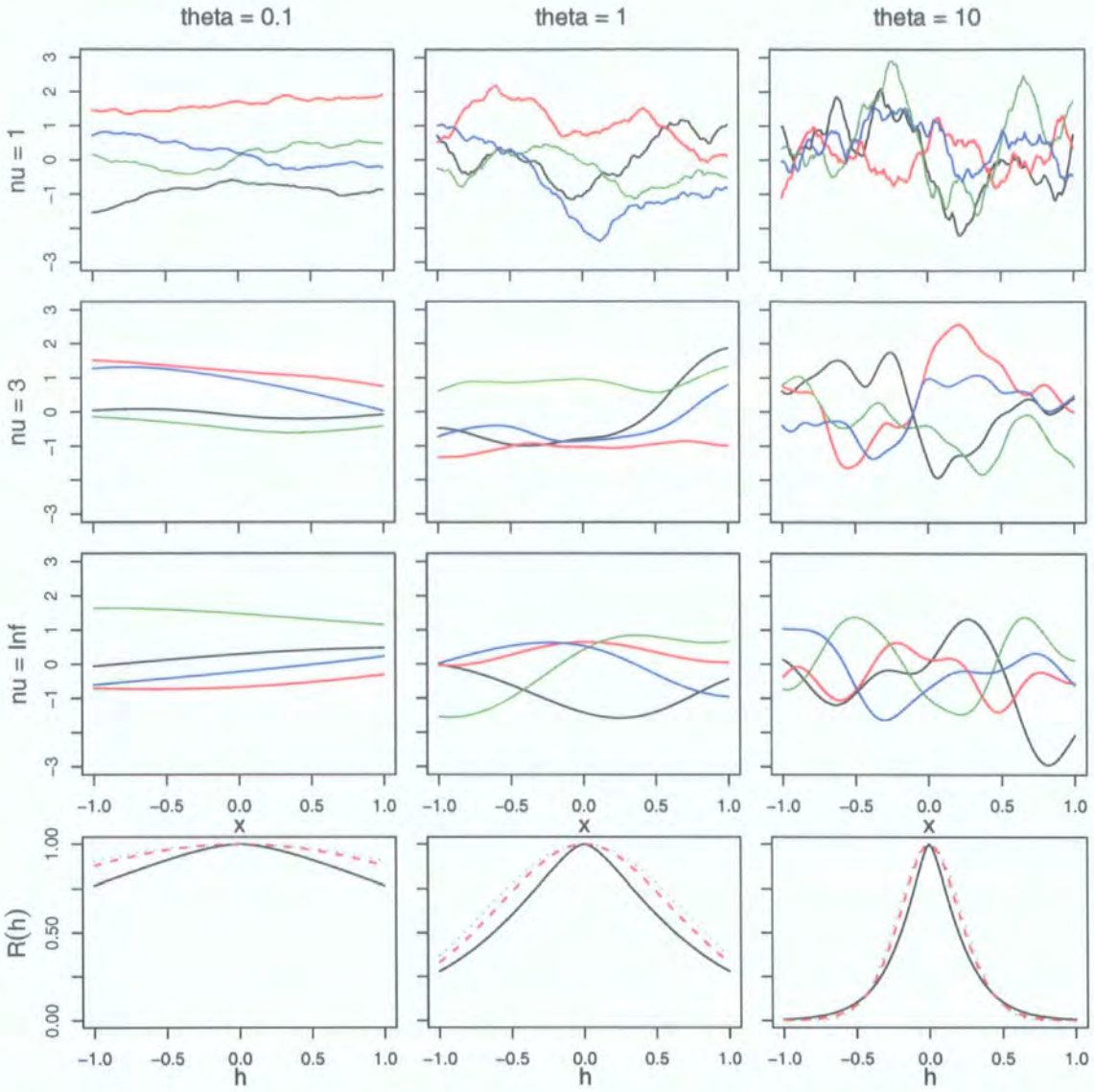


Figure 3.2: Realisations from a Gaussian process with Matérn Correlation function, $R_{\nu,\theta}(x)$, shown for different ν and θ values. The final row shows $R_{\nu,\theta}(h)$ with θ fixed at the corresponding column value and varying $\nu = 1$ (black), $\nu = 4$ (red), $\nu = \infty$ (green).

θ (move from left to right in the Figure), realisations become more wiggly, with an increasing number of turning points. The effect is purely one of scaling, however, with us effectively stretching the realisations sideways as we decrease θ. Comparing plots in the final row we see that, as we increase θ, correlations between pairs of points decrease. Increasing ν (moving from top to bottom in the Figure), we increase the differentiability of the realisations. This is most in evidence for $\nu = 1$ (top row) where we see that realisations are not differentiable no matter how 'smooth' the value

of θ. Moving from the second to third row takes us from third order differentiability to infinite differentiability (corresponding to the Gaussian case), although it is hard to see much difference by eye. One of main differences in the correlation functions (bottom row) is visible at $h = 0$. For $\nu = 1$ (black), it is clear that the correlation function is not differentiable at $h = 0$. In Chapter 4, where we discuss more closely differentiability of random processes, we see that the conditions of differentiability we develop do not hold anywhere if they do not hold at the origin. Again it is difficult to see much difference between $\nu = 4$ and $\nu = \infty$ although the former is four times differentiable at the origin and the latter infinitely differentiable.

The Power Exponential and Matérn families have natural product form generalisations, for multivariate $x = (x_1, \ldots, x_p)$, given by $r(x) = \prod_{i=1}^{p} r(x_i)$, where we are free to take as few or as many of the parameters to be indexed by i. Although the families describe within them a collection of covariance functions with a wide range of different behaviours, they are by no means the only possibilities; see Schlather (1999) for many more.

A possible extension to the model arises when there exists a subset x^* of x, termed the "active inputs", which are responsible for most of the systematic variation in $s(x)$. When this is the case, as in Craig et al. (1997), we take $g(x) = g(x^*)$ to depend only on active inputs. Furthermore, we split $\gamma(x)$ further so that

$$\gamma(x) = \epsilon(x^*) + \delta(x \setminus x^*) \tag{3.6}$$

In this representation, the ϵ-surface is a correlated error structure describing variation explained by active inputs but not picked up by the global effects, and the δ-surface, taken to be uncorrelated with the ϵ-surface, describes any variation explained by the set of remaining (non-active) inputs $x \setminus x^*$. The advantages in this representation arise when we believe that the δ-surface, which captures variation in non-active inputs, has variance, σ_δ^2, which is small. We discuss these advantages in Chapter 4.

3.3 Bayesian approach

A Bayesian approach lends itself well to the problem of building and refining an emulator for a simulator. From a Bayesian perspective, the emulator is interpreted as a representation of our beliefs about the relationships between inputs and outputs in the simulator. Starting with prior beliefs we can then update these beliefs as we observe simulator output at different input choices. Not only is this updating process very natural, but also our ability to build in any prior information we have about the simulator is potentially very useful, given that we can only expect to obtain a thin coverage of the input space.

Bayesian approaches, such as those in Currin et al. (1991) and Kennedy and O'Hagan (2001a,b), have also relied heavily on Gaussian processes for tractability. Assuming a Gaussian process prior for $s(\cdot)$,

$$s(\cdot) \sim N\left(m(\cdot),\, v(\cdot,\cdot)\right), \tag{3.7}$$

the posterior distribution for $s(\cdot)$, conditioned on the set $S = (s(X_1), \ldots, s(X_n))$ of observed function values (assuming for the time being that $m(\cdot)$ and $v(\cdot,\cdot)$ are known), is also a Gaussian process

$$s(\cdot)\,|S \sim N\left(m_S(\cdot),\, v_S(\cdot,\cdot)\right) \tag{3.8}$$

where

$$m_S(x) = m(x) + C_x^T V^{-1}(S - M) \tag{3.9}$$

$$v_S(x,x') = v(x,x') - C_x^T V^{-1} C_{x'} \tag{3.10}$$

$$C_x = \begin{pmatrix} v(x,X_1) \\ \vdots \\ v(x,X_n) \end{pmatrix}, \quad M = \begin{pmatrix} m(X_1)^T \\ \vdots \\ m(X_n)^T \end{pmatrix}, \tag{3.11}$$

$$V = \begin{pmatrix} v(X_1,X_1) & \ldots & v(X_1,X_n) \\ \vdots & & \vdots \\ v(X_n,X_1) & \ldots & v(X_n,X_n) \end{pmatrix}. \tag{3.12}$$

In general, $m(\cdot)$ and $v(\cdot,\cdot)$ are unknown, and in this case the approach has typically been to specify their form through a collection of hyper-parameters on which (3.7) is conditional and then set up a hierarchical model. For example, O'Hagan (1992) treats (3.7) as the first conditional factor in the following hierarchical model

$$P(s(\cdot)) = P(s(\cdot)|\beta,\sigma^2)P(\beta|\sigma^2)P(\sigma^2) \tag{3.13}$$

where β and σ^2 are hyper-parameters corresponding to

$$\mathrm{E}[s(x)|\beta,\sigma^2] = g(x)^T\beta \tag{3.14}$$

$$\mathrm{Cov}[s(x),s(x')|\beta,\sigma^2] = \sigma^2 r(x,x'). \tag{3.15}$$

The vector $g(x)^T = (g_1(x),\ldots,g_q(x))$ is a q-vector of known regressor functions and $M = G\beta$ in (3.9) with $G^T = (g(X_1),\ldots,g(X_n))$. Prior Normal and Inverse-χ^2 distributions are attached to β and σ^2 respectively

$$\beta \,|\, \sigma^2 \sim N(b_0, \sigma^2 B_0^{-1}), \tag{3.16}$$

$$\sigma^2 \sim s_0 \chi_{a_0}^{-2}. \tag{3.17}$$

where the latter is characterised by

$$p(\sigma^2) \propto \sigma^{-a_0} \exp^{-\frac{1}{2}\sigma^{-2}s_0}, \quad \sigma^2 \geq 0 \tag{3.18}$$

This yields posterior distribution for β,

$$\beta \,|\, (\sigma^2, S) \sim N(b, \sigma^2 B^{-1}), \tag{3.19}$$

where

$$b = B^{-1}(B_0 b_0 + G^T V^{-1} S), \tag{3.20}$$

$$B = B_0 + G^T V^{-1} G \tag{3.21}$$

and posterior for σ^2,

$$\sigma^2 \,|\, S \sim s\chi_a^{-2}, \tag{3.22}$$

where

$$a = a_0 + n - q, \tag{3.23}$$

$$s = s_0 + S^T \left(V^{-1} - V^{-1}G(G^TV^{-1}G)^{-1}G^TV^{-1}\right) S. \tag{3.24}$$

In general, further hyper-parameters will be introduced into (3.13) through specification of the correlation function $r(x, x')$. Kennedy and O'Hagan (2001a,b) consider the Gaussian and Matérn correlation functions and note that integrating out these hyper-parameters is only possible numerically using computationally intensive high-dimensional quadrature. Acknowledging that much of their subsequent methodology is computationally intensive, they conclude that a full Bayes analysis of these hyper-parameters will often not be possible and opt to 'derive plausible estimates of these parameters and then to act as if they were fixed'. Currin et al. (1991) use computationally demanding maximum likelihood methods to estimate hyper-parameters, but discover a further problem - that often there is not enough information in the data to distinguish between competing values of (σ, Θ) in the Gaussian covariance function. For a good discussion of the problems of maximum likelihood for estimating these parameters, see Ripley (1988). Estimating these parameters is something we will return to in Chapter 4.

3.4 Partial Beliefs: Bayes Linear Methods

Bayesian methodology provides a mathematically rigorous and philosophically sound framework to deal with the subjective aspects inherent in real-world situations involving uncertainty. However, in almost all practical situations, it is beset by two main problems. Firstly, although probability distributions are a natural medium for representing quantities of uncertainty, they do not necessarily transfer to the representation of an individual's subjective beliefs. In particular, individuals can rarely justify distributional assumptions for their beliefs (to see this, consider a single, scalar quantity and consider your beliefs about each of its moments). The parametric families employed will be chosen for convenience and, although in simple situations they may represent a good approximation to actual beliefs, in more complicated situations, results can potentially be sensitive to specifications which are

not strongly held. The second issue is computational: as we saw in Section 3.3, posterior distributions are often not available in closed form and the methodology must either be compromised or computationally-intensive numerical integration methods employed, in turn limiting the scale of problems that can be dealt with.

Motivated by these two issues, the Bayes Linear methodology aims to enable rigorous statement and updating of partial belief specifications within a framework which also does not fall prey to the same computational pitfalls of a full Bayes analysis. In doing this, it effectively kills two birds with one stone: removing full distributional assumptions whose connection to actual beliefs is tenuous at best and, in doing so, throwing off the computational millstone attached to analysing these 'beliefs'.

The Bayes Linear approach treats expectation, rather than probability, as a primitive for expressing our beliefs about random quantities (See de Finetti (1974) for a careful treatment of expectation as a primitive and Goldstein (1999) for a survey of the Bayes Linear approach). In particular, for a collection, $\mathcal{C} = (C_1, C_2, \ldots)$, of random quantities, we specify directly:

1. the expectation, $\mathrm{E}[C]$, for each element C in $\mathcal{C}$
2. the variance, $\mathrm{Var}[C]$, for each element C in $\mathcal{C}$
3. the covariance, $\mathrm{Cov}[C, C']$, for each pair of elements (C, C') in $\mathcal{C}$

Formally, we define the *Belief Structure*, $[\mathcal{C}]$, for $\mathcal{C}$ to be the inner product space formed from:

1. the space of linear finite combinations of the elements of $\mathcal{C}$ and the unit constant C_0, together with:
2. the inner product $\langle \cdot, \cdot \rangle$ and norm $|| \cdot ||$ for $C, C' \in \mathcal{C}$ defined by

$$\langle C, C' \rangle = \mathrm{Cov}[C, C'] \tag{3.25}$$

$$||C||^2 = \mathrm{Var}[C] \tag{3.26}$$

To complete the definition, we define the inner product space over the closure of the equivalence classes of random quantities which differ by a constant. This

enables us to standardise each quantity by subtracting its prior expectation so that, in particular, all constant terms become equivalent to zero and $||C|| = 0$ if and only if $C = 0$ as required.

Suppose we observe data, $\mathcal{D} = (D_1, \ldots, D_s)$, and let $\mathcal{B} = \mathcal{C} \backslash \mathcal{D}$ be the collection of remaining unobserved quantities in $\mathcal{C}$ about which we wish to revise our beliefs given the data. Then *the adjusted expectation of $\mathcal{B}$ given $\mathcal{D}$* is the element $E_{\mathcal{D}}[\mathcal{B}] \in [\mathcal{D}]$ that minimises $||\mathcal{B} - \mathrm{E}_{\mathcal{D}}[\mathcal{B}]||$. In other words, $E_{\mathcal{D}}[B]$, for $B \in \mathcal{B}$, is the linear combination of the data (including constant term D_0), $E_{\mathcal{D}}[B] = \sum_{j=0}^{s} c_j D_j$, which minimises

$$L = ||B - \sum_{j=0}^{s} c_j D_j|| \tag{3.27}$$

over the space of possible c.

Squaring both sides of (3.27), differentiating with respect to components of c, equating to zero and solving the resulting matrix equations, we obtain

$$\mathrm{E}_D[B] = \mathrm{E}[B] + \mathrm{Cov}[B, D]\mathrm{Var}[D]^{-1}(D - \mathrm{E}[D]) \tag{3.28}$$

From (3.28) we see that the rule has several intuitively appealing properties. Firstly, we note that $E_B[B] = B$ but also that the influence of the data in the adjustment is larger the larger the prior correlation between B and D, the smaller the prior data variance, $\mathrm{Var}[D]$, and the greater the distance between the prior expectation, $\mathrm{E}[D]$, and the observed value D.

Geometrically, $E_{\mathcal{D}}[B]$ is the orthogonal projection of B onto $[\mathcal{D}]$. The adjusted variance of B given $\mathcal{D}$ is the squared orthogonal distance from B to $\mathcal{D}$, given by

$$\begin{aligned} \mathrm{Var}_D[B] &= ||B - \mathrm{E}_D[B]||^2 \\ &= \mathrm{Var}[B] - \mathrm{Cov}[B, D]\mathrm{Var}[D]^{-1}\mathrm{Cov}[D, B] \end{aligned} \tag{3.29}$$

Eqn. (3.29) offers intuitive properties in that the greater the prior correlation between B and D, the greater the amount of variance reduced by observing D.

Similarly, the *covariance of B and B' adjusted by D* is given by

$$\begin{aligned} \mathrm{Cov}_D[B, B'] &= \mathrm{E}[(B - \mathrm{E}_D[B])(B' - \mathrm{E}_D[B'])] \\ &= \mathrm{Cov}[B, B'] - \mathrm{Cov}[B, D]\mathrm{Var}[D]^{-1}\mathrm{Cov}[D, B'] \end{aligned} \tag{3.30}$$

It is worth noting that (3.28) and (3.29) correspond to (3.9) and (3.10), the conditional mean and variance obtained in the full Bayes analysis based on a Gaussian Process with known mean and covariance function.

3.5 Bayes Linear emulators

We follow the approach of Craig et al. (2001, 1996, 1997, 1998) in only requiring second order beliefs to be specified and then using Bayes Linear methods to update these beliefs. In general B denotes a vector of quantities about which we have beliefs and D an observed data vector related in some way (as defined by its covariance structure) to B. For our purposes, B is replaced by the simulator, $s(x)$, and D by $S = s(X)$, the set of simulator output observed at $X = (X_1, \ldots, X_n)$.

Our reasons for adopting the Bayes Linear approach in this thesis are the two given at the beginning of Section 3.4: firstly, the simulators we are concerned with are highly complex and consequently very difficult to elicit full belief structures for and, secondly, the approach allows certain largely intractable calculations to be performed without the computationally intensive methods required in a full Bayes analysis. Furthermore, in the next chapter, where we consider extending the methodology to allow for observation of derivatives, deriving the required joint distributions in a full Bayes analysis further compounds the heavy reliance on a Gaussian process framework for tractability.

Chapter 4

Adjusting emulators based on derivatives

In this chapter, we consider the situation in which derivatives of the simulator are available and show that the methodology described in Chapter 3 can be extended quite naturally to allow for inclusion of such derivatives. We begin with some necessary background on random processes and their derivatives, before developing the extensions to the methodology in Chapter 3, which allow us to update beliefs about the simulator based on derivatives when they are available. Finally, we perform some theoretical calculations to get an idea about how much extra information we can hope to gain from including derivatives.

4.1 Random Processes and derivatives

In this section, we summarise some basic ideas concerning random processes and, in particular, those concepts related to the differentiation of such processes. The discussion is based on Yaglom (1986), where more details can be found.

Let $Y(x)$, x scalar, be a random process on probability space χ. Then $Y(x)$ is a random process of second order if the expectation $m(x) = \mathrm{E}[Y(x)]$ and covariance $C(x, x') = \mathrm{Cov}[Y(x), Y(x')]$ exist. A process $Y(x)$ is said to be mean-squared differentiable at $x = a$ if there exists a random variable $Y^{(1)}(a)$ with $\mathrm{E}[Y^{(1)}(a)] < \infty$

such that

$$\lim_{h\to 0} \mathrm{E}\left|\frac{Y(a+h)-Y(a)}{h} - Y^{(1)}(a)\right|^2 = 0 \tag{4.1}$$

The process $Y^{(1)}(x)$ exists on an interval, $\mathcal{I}$, if and only if the derivatives

$$\frac{dm(x)}{dx}, \quad \frac{\partial C(x,x')}{\partial x}, \quad \frac{\partial C(x,x')}{\partial x'} \tag{4.2}$$

exist on $\mathcal{I}$. When this is the case, the quantities in (4.2) correspond to $\mathrm{E}[Y^{(1)}(x)]$, $\mathrm{Cov}[Y^{(1)}, Y]$ and $\mathrm{Cov}[Y, Y^{(1)}]$ respectively.

Higher order derivatives can then be defined by a simple inductive argument so that, for example, $Y^{(2)}(x)$ is the mean squared derivative of $Y^{(1)}(x)$ and so on. This leads to analogous conditions that a process $Y(x)$ has nth mean square derivative if and only if the derivatives

$$\frac{d^i}{dx^i}m(x), \quad \frac{\partial^{i+j}}{\partial x^i \partial x'^j}C(x,x')$$

exist for all $i, j \leq n$. Moreover, when this is the case,

$$\mathrm{E}[Y^{(i)}(x)] = \frac{d}{dx^i}m(x) \tag{4.3}$$

$$\mathrm{Cov}[Y^{(i)}(x), Y^{(j)}(x')] = \frac{\partial^{i+j}}{\partial x^i \partial x'^j}C(x,x') \tag{4.4}$$

It is similarly straightforward to generalise the ideas of mean square differentiability to multivariate x.

Before we proceed any further, let us first stop and examine the definition of mean-square differentiability given in (4.1). Note that the mean-square differentiability of the process $Y(x)$ for all x does not mean that all realisations $y(x)$ of this process will be differentiable. Rather the requirement is that, for a given x, the probability that the process is non-differentiable in the interval $[x, x+h]$, for h small, is very low so that the mean square of $Y^{(1)} - \frac{Y(x+h)-Y(x)}{h}$ is also small. In fact, the realisation $y'(x)$ of the derivative $Y'(x)$ of a differentiable process $Y(x)$ may be a discontinuous function of x. On the other hand, the process $Y(x)$ may also be non-differentiable when all its realisations $y(x)$ have continuous realisations $y'(x)$ (Yaglom, 1986, provides examples of both).

In practice, however, this turns out to be something of a technicality and, for those differentiable stationary processes suitable for practical situations, realisations $y(x)$ turn out to be differentiable and the functions $y'(x)$ coincide with the realisations of the process $Y'(x)$. This is the case for the Matérn correlation function for which, as we remarked in Chapter 3, derivatives of realisations of a process with smoothness parameter ν are differentiable to order $[v]-1$. It also true for the Gaussian correlation function (to see this, recall that the Gaussian is simply the limit of the Matérn as $\nu \to \infty$). The coincidence of derivatives and realisations is a crucial one for our needs in the next section, where we will be concerned with updating beliefs about a process based on derivatives realised from that process.

4.2 Beliefs about Derivatives

Specifying an emulator for a function induces an emulator for the derivatives of the function. In this section, we derive the covariances between a function and its derivatives and discuss how we might exploit these implications to improve our specification and refinement of the emulator.

For a univariate process, $s(x)$, with covariance function, $v(x,x')$, the order of integration and differentiation can be swapped provided the indicated derivatives exist so that, for example,

$$\mathrm{Cov}\left[\partial^m s(x)/\partial x_i^m,\ \partial^n s(x')/\partial x_j'^n\right] = \partial^{m+n} v(x,x')/\partial x_i^m \partial x_j'^n. \tag{4.5}$$

Recall the emulator model specified by (3.1) and (3.6)

$$s(x) = Bg(x^*) + \epsilon(x^*) + \delta(x \setminus x^*). \tag{4.6}$$

If x_i and x_j belong to the set, x^*, of active inputs, then provided $g(x)$ is differentiable, we have

$$\begin{aligned}
v(x,x') &= g(x)^T \mathrm{Var}[B] g(x') + \mathrm{Cov}[\epsilon(x),\ \epsilon(x')] + \mathrm{Cov}[\delta(x),\ \delta(x')] \\
\partial v(x,x')/\partial x_i &= \partial g(x)/\partial x_i \mathrm{Var}[B] g(x') + \partial/\partial x_i \mathrm{Cov}[\epsilon(x),\ \epsilon(x')] \\
\partial^2 v(x,x')/\partial x_i \partial x_j' &= \partial g(x)/\partial x_i \mathrm{Var}[B] \partial g(x')/\partial x_j' + \partial^2/\partial x_i \partial x_j' \mathrm{Cov}[\epsilon(x),\ \epsilon(x')]
\end{aligned} \tag{4.7}$$

If either x_i or x_j does not belong to x^*, but rather is inactive, then differentiating (4.6) yields

$$\begin{aligned} \partial v(x,x')/\partial x_i &= \partial/\partial x_i \mathrm{Cov}[\delta(x),\, \delta(x')] \\ \partial^2 v(x,x')/\partial x_i \partial x'_j &= \partial^2/\partial x_i \partial x'_j \mathrm{Cov}[\delta(x),\, \delta(x')] \end{aligned} \tag{4.8}$$

For the ϵ-surface, the Gaussian covariance kernel, given for multivariate x by

$$\mathrm{Cov}[\epsilon(x),\, \epsilon(x')] = \exp^{-(x-x')^T \Theta (x-x')}, \quad \Theta = \mathrm{diag}(\theta_1, \ldots, \theta_p), \quad \theta_i > 0, \tag{4.9}$$

is infinitely differentiable and so the interchange of order of differentiation and integration is allowed. Differentiating (4.9) and making the interchange of order explicit, we obtain

$$\begin{aligned} \mathrm{Cov}[\partial\epsilon(x)/\partial x_i,\, \epsilon(x')] &= -2\theta_i(x_i - x'_i)\mathrm{Cov}[\epsilon(x),\, \epsilon(x')] \\ \mathrm{Cov}[\partial\epsilon(x)/\partial x_i,\, \partial\epsilon(x')/\partial x'_i] &= 2\theta_i \left(1_{(i=j)} - 2\theta_j(x_i - x'_i)(x_j - x'_j)\right) \mathrm{Cov}[\epsilon(x),\, \epsilon(x')] \end{aligned} \tag{4.10}$$

where $1_{(i=j)} = 1$ if $i = j$ and zero otherwise. An immediate consequence of the equations in (4.10) is the following three equations:

$$\mathrm{Cov}[\partial\epsilon(x)/\partial x_i,\, \epsilon(x)] = 0 \tag{4.11}$$

$$\mathrm{Cov}[\partial\epsilon(x)/\partial x_i,\, \partial\epsilon(x)/\partial x_j] = 0 \text{ for } i \neq j \tag{4.12}$$

$$\mathrm{Var}[\partial\epsilon(x)/\partial x_i] = 2\theta_i\sigma_\epsilon^2 \tag{4.13}$$

There are a couple of interesting points to note from this second set of equations. Firstly, we see from (4.11) and (4.12) that, at any given point, the separate pieces of information given by the function value and each of the first order derivatives are all orthogonal to each other in terms of what they tell us about the function at that point. Secondly, we see from (4.13) that the variability of the derivative with respect to x_i is related linearly to θ_i. This suggests that derivatives may be useful in estimating Θ, something we will return to in Section 4.5. The top left hand panel in Figure 4.1 plots the covariance between function values and derivatives derived in (4.10) for x scalar. We see the effect of the multiplicative distance term, which appears in the expression for $\mathrm{Cov}[\epsilon(x),\, d\epsilon(x')/dx']$, is that the covariance between $\epsilon(x)$ and a derivative observation, $d\epsilon(x')/dx'$, does not decrease as quickly

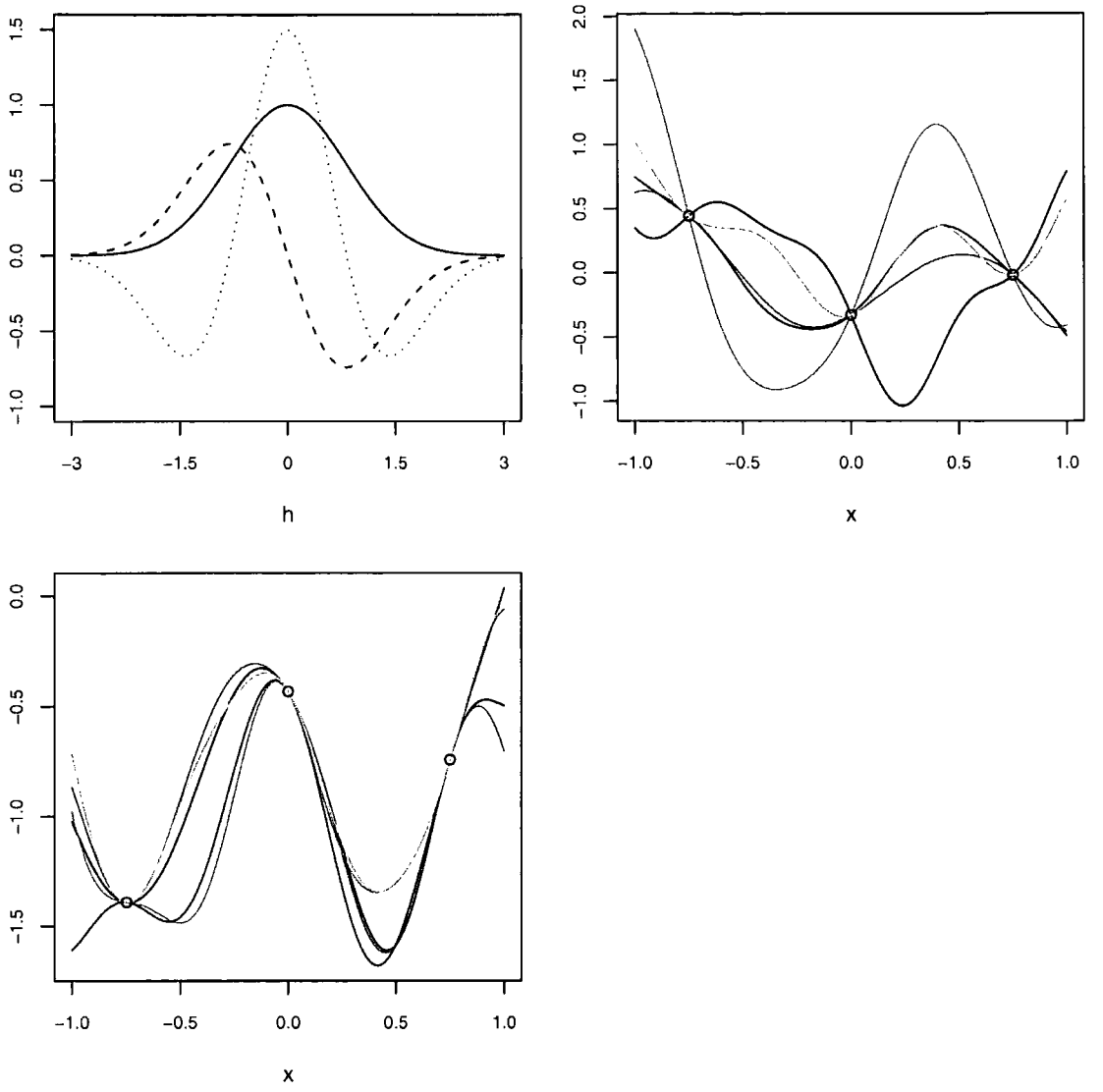


Figure 4.1: (Top left) Covariance between function values and derivatives for the Gaussian covariance kernel in one dimension with hyper-parameters $\sigma_\epsilon = 1$, $\theta = 1.5$ plotted against the distance between observations, $h = ||x - x'||$. The thick lines shows $\mathrm{Cov}[\epsilon(x), \epsilon(x')]$, the dashed line shows $\mathrm{Cov}[\epsilon(x), d\epsilon(x')/dx']$ and the dotted line shows $\mathrm{Cov}[d\epsilon(x)/dx, d\epsilon(x')/dx']$. The top right and bottom left plots show the constraining effects of function values and additionally with derivatives respectively.

as $h = ||x - x'||$ increases. We also see there are two turning points for the derivative covariance, which we can show are at $h_i = \pm\sqrt{3/\theta_i}$, at which the correlation function attains its highest negative value. We can easily generate expressions for covariances of higher order derivatives should we wish. In Appendix G we present a simple way

to do this for the Gaussian covariance kernel.

Note that in (4.9), Θ can be generalised to be any positive definite matrix so that its off-diagonal terms are not necessarily zero. Taking Θ to be diagonal tends to be the method of choice, however, and is convenient because it admits a product form for the covariance function. (O'Hagan, 1998, shows the diagonal form corresponds to a Markov-type assumption on $\epsilon(x)$).

Modelling the δ-surface in (4.6) similarly with a correlated covariance structure would enable us to include derivatives of non-active inputs in an identical fashion. However, we choose to take the following structure

$$\mathrm{Cov}[\delta(x),\, \delta(x')] = \begin{cases} \sigma_\delta^2 \text{ for } x = x', \\ 0 \text{ for } x \neq x'. \end{cases} \tag{4.14}$$

Eqn. (4.14) represents a combination of two simplifying assumptions for the δ-surface part of our emulator model. Firstly, we assume that any runs we make on the simulator are sufficiently far apart in the space of non-active inputs for the δ residuals of any fit to be effectively uncorrelated. Secondly, we reduce the dimensionality of $s(x)$ by considering it as a function only of active inputs so we treat the variance of the δ-surface, σ_δ^2, as the loss in precision attributable to the dimension reduction, being the irreducible lower bound on our uncertainty at any given x^*. An immediate consequence of the latter is that we evaluate our emulator beliefs, $s(x)$, over a grid of dimension $\dim(x^*)$ rather than $\dim(x)$ leading to a large reduction in the computational burden of required calculations, enabling us to obtain a reasonable coverage of the space provided the number of active variables is small. The price we pay for these large computational savings is that derivatives with respect to inactive inputs can not be included in our updating since $s(x)$ is not differentiable with respect to these arguments. However, this will be a price worth paying if our assumption that σ_δ^2 is small holds true, since then the simulator will be flat in the inactive input dimensions and, since any offset from zero will have been absorbed by the systematic part of the model, these derivatives will not be informative.

4.3 Discussion

In a Bayesian context, the emulator is interpreted as a representation of our beliefs about the relationships between inputs and outputs. Since (4.7) specifies joint beliefs about derivatives and function values, we can exploit this underlying covariance structure and make use of observations of derivatives using the Bayes Linear formulae derived in Chapter 3:

$$\begin{aligned} \mathrm{E}_D[B] =& \mathrm{E}[B] + \mathrm{Cov}[B, D]\mathrm{Var}[D]^{-1}(D - \mathrm{E}[D]) \\ \mathrm{Var}_D[B] =& \mathrm{Var}[B] - \mathrm{Cov}[B, D]\mathrm{Var}[D]^{-1}\mathrm{Cov}[D, B] \end{aligned} \tag{4.15}$$

Since we are interested in the effects of observing derivatives, we define the operator $\nabla_x = (\partial/\partial x_1, \ldots, \partial/\partial x_p)$ which, when applied to a quantity, returns the partial derivatives of that quantity with respect to components of the vector $x = (x_1, \ldots, x_p)$. Hence, $\nabla_x S$ is the collection of first order derivatives of $s(x)$ at $X = (X_1, \ldots, X_n)$. In particular, we will compare updating based on (4.15) for the cases $D = S$ and $D = (S, \nabla_{x^*} S)$ where, for the latter, we update based only on derivatives with respect to variables in the set of active inputs, x^*, because it is these derivatives which we expect to be most informative and doing so allows us to make simplifying assumptions in our treatment of the δ-surface, discussed in section 6.3, which reduce computational burdens.

We do not know of any systematic application of functional derivatives for Bayesian emulation in the literature, although the potential to incorporate derivative information was pointed out, in principle, by O'Hagan (1992) who considered first derivatives of Gaussian processes in the context of numerical analysis. Theoretical considerations relating to design, also when first derivatives are available, are tackled in Mitchell et al. (1993, 1994), who derive asymptotic results in the case that intersite correlations become progressively weaker. Santner et al. (2003, p. 107) also give a simple example to demonstrate the potential in which derivative information is shown to lead to an improved predictive performance of the emulator.

Specifying beliefs about a function induces beliefs about the derivatives of that function. In practice, for the inclusion of derivative information to be worthwhile, we need an efficient and relatively inexpensive method for generating derivatives.

In Chapter 5, we introduce a cheap and accurate method for obtaining derivatives for compartmental simulators, such as that introduced in Chapter 2. However, even in situations where observations on derivatives are not available, it may well be insightful to derive the belief representations (4.7), (4.10) and to reflect on whether they are in agreement with those prior beliefs that are actually held.

As we remarked in Chapter 3, in the important special case that the prior is a Gaussian Process with known mean and covariance function, (3.28) and (3.30) correspond to the conditional mean and variance obtained in the full Bayes analysis. This is also true in the case that D contains derivatives because a function which is normally distributed has a joint normal distribution with its derivatives (see e.g. O'Hagan, 1978). However, whilst the full distributional treatment of derivatives, required in a full Bayes analysis, has only been demonstrated to be tractable within the Gaussian process framework, a Bayes Linear analysis remains tractable without any such restriction, requiring only that the covariance function is differentiable to the required order.

A final remark is that it is prudent to be wary of the dangers of misspecification of the variance function for $s(\cdot)$. Seemingly similar covariance functions can often give rise to quite different inferences based on simulator evaluations and we would expect this to even more the case when derivative information is included in the analysis since its inclusion amounts to an increased reliance on the model assumptions, on one level at least. Whilst squeezing more from the model assumptions is desirable, this should of course be tempered by placing emphasis on precautionary diagnostics.

4.4 Theoretical calculations of resolved variance

In this section, we consider a process with Gaussian covariance function and compare variance resolved by observing a function value with that resolved by observing the function value and the collection of first-order derivatives of the function. We then consider a collection of design points and derive a result for the trade-off between function values and derivatives - in terms of resolved variance - when the design points are effectively independent. We relate bounds on the Θ covariance parameter

to bounds on the validity of the independence approximations. Finally, we conclude with a discussion of our results and possible extensions.

4.4.1 Variance resolved by a single input point

Suppose we have a set $D = (D_1, \ldots, D_n)$ of observations on aspects of a process $\epsilon(x)$ at a set of n design points $X = (X_1, \ldots, X_n)$. Then the Resolved Variance $\mathrm{RVar}_D[\epsilon(x)]$ of $\epsilon(x)$ by D is

$$\begin{aligned} \mathrm{RVar}_D[\epsilon(x)] &= \mathrm{Var}[\epsilon(x)] - \mathrm{Var}_D[\epsilon(x)] \\ &= \mathrm{Cov}[\epsilon(x), D]\mathrm{Var}[D]^{-1}\mathrm{Cov}[D, \epsilon(x)] \end{aligned} \tag{4.16}$$

where $\mathrm{Var}_D[\epsilon(x)]$ is the variance of $\epsilon(x)$ adjusted by D, defined in (3.29).

Recall the relevant expressions for the Gaussian covariance kernel, given in (4.10),

$$\mathrm{Cov}[\epsilon(x), \epsilon(x')] = \sigma_\epsilon^2 \exp^{-(x-x')^T\Theta(x-x')} \tag{4.17}$$

$$\mathrm{Cov}[\partial\epsilon(x)/\partial x_i,\, \epsilon(x')] = -2\theta_i(x_i - x_i')\mathrm{Cov}[\epsilon(x),\, \epsilon(x')] \tag{4.18}$$

$$\mathrm{Cov}[\partial\epsilon(x)/\partial x_i,\, \partial\epsilon(x')/\partial x_i'] = 2\theta_i\left(1_{(i=j)} - 2\theta_j(x_i - x_i')(x_j - x_j')\right)\mathrm{Cov}[\epsilon(x),\, \epsilon(x')] \tag{4.19}$$

where $1_{i=j} = 1$ if $i = j$ and zero otherwise. Suppose we observe $S = \epsilon(0)$, the function value at the origin. Then substituting for (4.17) into (4.16), we find

$$\mathrm{RVar}_S[\epsilon(x)] = \sigma_\epsilon^2 \exp^{-2x^T\Theta x} \tag{4.20}$$

Now suppose instead that we observe $\nabla_x S = \left(\frac{\partial\epsilon(0)}{\partial x_1}, \ldots, \frac{\partial\epsilon(0)}{\partial x_p}\right)$, the set of p first order derivatives of $\epsilon(x)$, again at the origin. Then we see from (4.17 - 4.19) that $\mathrm{Var}[\nabla_x S] = 2\Theta\sigma_\epsilon^2$ which is easily invertible since Θ is diagonal. Hence, plugging this into (4.16) together with the covariance terms, $\mathrm{Cov}[\epsilon(x), \nabla_x S]$, - given in (4.18) - the resolved variance is found to be

$$\mathrm{RVar}_{\nabla_x S}[\epsilon(x)] = 2x^T\Theta x \times \sigma_\epsilon^2 \exp^{-2x^T\Theta x} \tag{4.21}$$

Recalling our remarks in light of (4.11), that the function value and derivatives are orthogonal at any given point, we have

$$\mathrm{RVar}_{(S,\nabla_x S)}[\epsilon(x)] = \mathrm{RVar}_S[\epsilon(x)] + \mathrm{RVar}_{\nabla_x S}[\epsilon(x)] \tag{4.22}$$

$$= \sigma_\epsilon^2\left(1 + 2x^T\Theta x\right)\exp^{-2x^T\Theta x} \tag{4.23}$$

Now suppose $\tilde{x} \subseteq x$ with $\dim(\tilde{x}) = \tilde{p} \leq p$ and reorder the components of x so that $\tilde{x} = (x_1, \ldots x_{\tilde{p}})$. Then the analogous expression for $\nabla_{\tilde{x}} S$ to (4.23) is

$$\mathrm{RVar}_{(S,\nabla_{\tilde{x}}S)}[\epsilon(x)] = \sigma_\epsilon^2 \left(1 + 2\tilde{x}^T \tilde{\Theta} \tilde{x}\right) \exp^{-2x^T \Theta x} \tag{4.24}$$

where $\tilde{\Theta} = \mathrm{diag}(\theta_1, \ldots, \theta_{\tilde{p}})$. From (4.20 - 4.24) we see that, as we would expect, we resolve more variance the more derivative information we include; the question is *how much more*? Specifically, we are interested in the trade-off between derivative information and function information. This is relevant because, in the case of a computer simulator, the code will take longer to run if it computes derivative information as well as function information; hence knowing how much including derivatives allows us to decrease the number of input points, whilst still maintaining the same level of variance reduction, tells us how high the computational overhead of computing derivatives can be and it be worth our while to compute them.

Clearly, "how much we learn" has yet to be defined. For the purposes of this work, we consider the *Integrated Resolved Variance* for D

$$\mathrm{IRVar}_D = \int_\chi \mathrm{RVar}_D[\epsilon(x)] dx \tag{4.25}$$

which is simply the variance resolved by D, integrated over the p-dimensional bounded design space $\chi = [-1, 1]^p$. The quantity is analogous to the *Integrated Mean Squared Error* of Sacks et al. (1989) and the integral of the posterior variance in the Full Bayes approach of O'Hagan (1992); both authors developing designs based on minimising its value.

Setting $\sigma_\epsilon^2 = 1$, without loss of generality, we see from (4.20) that

$$\mathrm{IRVar}_S = \prod_{i=1}^{p} \int_{-1}^{1} \exp^{-2\theta_i x_i^2} dx_i \tag{4.26}$$

Integrating (4.23),

$$\mathrm{IRVar}_{(S,\nabla_{\tilde{x}}S)} = \mathrm{IRVar}_S + \mathrm{IRVar}_{\nabla_{\tilde{x}}S} \tag{4.27}$$

so that the extra information gleaned from $\tilde{p}$ derivatives is given by

$$\begin{aligned} \mathrm{IRVar}_{\nabla_{\tilde{x}} S} &= 2\int_{\chi} \tilde{x}^T \tilde{\Theta} \tilde{x} \exp^{-2x^T \Theta x}\, dx \\ &= \int_{\chi} dx \sum_{i=1}^{\tilde{p}} 2\theta_i x_i^2 \prod_{j=1}^{p} \exp^{-2\theta_j x_j^2} \\ &= \sum_{i=1}^{\tilde{p}} 2\theta_i \prod_{j=1}^{p} \int_{-1}^{1} x_i^2 \exp^{-2\theta_j x_j^2}\, dx_j \end{aligned} \tag{4.28}$$

For each term in the sum, we have a product of $p-1$ integrals identical to those in the product for IRVar_S in (4.26), corresponding to the cases $i \neq j$, and a final integration corresponding to the case $i = j$ for which we can use integration by parts. Letting $u = x_i$ and $v' = x_i e^{-2\theta_i x_i^2}$, we have $u' = 1$ and $v = -\frac{1}{4\theta_i} e^{-2\theta_i x_i^2}$ and so

$$\int_{-1}^{1} x_i^2 e^{-2\theta_i x_i^2} dx_i = -\frac{2}{4\theta_i} e^{-2\theta_i} + \frac{1}{4\theta_i} \int_{-1}^{1} e^{-2\theta_i x_i^2} dx_i \tag{4.29}$$

The integral term in (4.29) contributes to the product an integral of the same form as the other components, multiplied by a factor of $1/4\theta_i$. If we divide the first term in (4.29) by $\int_{-1}^{1} \exp^{-2\theta_i x_i^2} d_{x_i}$, then we can factor the full product of p integrals, which is equal to IRVar_S, from both terms in (4.29) and take it outside the sum in (4.28) to obtain

$$\mathrm{IRVar}_{\nabla_{\tilde{x}} S} = \mathrm{IRVar}_S \times \sum_{i=1}^{\tilde{p}} 2\theta_i \left(\frac{-2e^{-2\theta_i}}{4\theta_i \int_{-1}^{1} e^{-2\theta_i x_i^2} dx_i} + \frac{1}{4\theta_i} \right) \tag{4.30}$$

$$= \mathrm{IRVar}_S \times \left(\frac{\tilde{p}}{2} - \sum_{i=1}^{\tilde{p}} \frac{e^{-2\theta_i}}{\int_{-1}^{1} e^{-2\theta_i x_i^2} dx_i} \right) \tag{4.31}$$

Note that the terms inside the sum tend to $1/2$ as $\theta \longrightarrow 0$ and can be made arbitrarily small by taking θ_i suitably large (since, as $\theta_i \longrightarrow \infty$, the integral in the denominator tends to $\sqrt{\pi/2\theta_i}$ which goes to zero much slowly than the numerator). Hence, as long as variation described by ϵ is reasonably localised (so that the variance resolved by observing function value and derivatives is essentially zero at the boundaries of the design space χ), then a single scalar derivative contributes half that of a function value to the integrated resolved variance. That is

$$\mathrm{IRVar}_{\nabla_{\tilde{x}} S} = \mathrm{IRVar}_S \times \frac{\tilde{p}}{2} \quad \text{for } \theta_i \text{ large.} \tag{4.32}$$

Hence observing the function value and set of p first order derivatives of $s(x)$ resolves the same amount of variance as observing two function values for $p = 2$ and resolves more variance for $p \geq 3$. In order to get an idea of how large the θ_i need to be realistically, if $\min_i \theta_i > 1.5$, then this corresponds to $\sigma^2 < 1/6$ in the standard Gaussian density $e^{-\frac{1}{2\sigma^2}x^2}$ and it is easy to show numerically that the terms in the sum in (4.31) are all less than 0.05 so that

$$\text{IRVar}_{\nabla_{\tilde{x}} S} > 0.45\tilde{p} \times \text{IRVar}_S \quad \text{for } \min_i \theta_i > 1.5. \tag{4.33}$$

4.4.2 Several independent design points

We are interested in generalising (4.33) to the case of several design points, $X_1, \ldots, X_n$, at which we observe some information $D = (D_1, \ldots, D_n)$. Suppose that there exist sub-regions of χ, $\mathcal{R}_1, \ldots, \mathcal{R}_n$, each centered on its respective design point such that $\int_{\mathcal{R}} \text{RVar}_{D_i} \approx \int_{\chi} \text{RVar}_{D_i}$. Then condition (4.33) is satisfied for each D_i since the variance resolved by D_i is outside of $\mathcal{R}_i \subset \chi$ is negligible. If no two regions overlap, the information at any design point is effectively independent from the information at the remaining design points (the resolved variance at the midpoint of the straight line between any two design points will be approximately zero) and

$$\int_{\chi} \text{RVar}_D(x)dx \approx \sum_{i=1}^{n} \int_{\mathcal{R}_i} \text{RVar}_{D_i}(x)dx. \tag{4.34}$$

Hence, since the covariance between two points depends only on the distance between them, the ratio of the variance resolved by the set, S_n, of function evaluations at design points $(X_1, \ldots, X_n)$ and the set, $(S_m, \nabla_x S_m)$, of function values and first order derivatives observed at design points $(X_1, \ldots, X_m)$, is

$$\frac{\text{IRVar}_{S_m, \nabla S_m}}{\text{IRVar}_{S_n}} = \frac{m(2+p)}{2n} \text{ for } \theta_i \text{ large.} \tag{4.35}$$

To get an idea about how large the θ_i need to be, consider the rectangular subregion, $\mathcal{R} = [-L_1, L_1] \times \cdots \times [-L_p, L_p]$, centered on some design point. Then, by a linear transformation of x_i, it is straightforward to show that the resolved variance, integrated over the subregion, $\mathcal{R}$, is identical to (4.31) but with θ_i replaced by $\theta_i L_i^2$. Hence, the condition becomes

$$\text{IRVar}_{\nabla_{\tilde{x}} S} > 0.45\tilde{p} \times \text{IRVar}_S \iff \theta_i > \frac{6}{4L_i^2} \text{ for all } i. \tag{4.36}$$

This is a powerful result since it tells us that, provided we have some condition on the smallest θ_i (for example, $\min_i \theta_i > 1.5$), then we can pack design points close together in dimensions with large θ_i values and our approximations will still be valid.

Hence, assuming that information at design points is reasonably independent, (4.35) tells us that we can expect to be able to reduce the number of design points by a factor of $\frac{2}{2+p}$ if we observe p derivatives at each location whilst still achieving the same (integrated) reduction in variance. A few comments are worth making at this point.

Firstly, typically our emulator model will contain some global terms, not included in our calculation to make the analysis tractable, and so we should emphasise that the factor $\frac{2}{2+p}$ relates only to the ϵ-surface (with Gaussian covariance function). We can think of this as the trade-off factor once most of the variation in global terms has been resolved. Note that the analysis would remain tractable if we were to include a constant global term, b since this would not affect covariances relating to derivatives in the data variance matrix and the variance matrix of observed function values would comprise $\sigma_b^2 + \sigma_\epsilon^2$ terms on the diagonal and σ_b^2 terms on the off-diagonal, an inverse to which is available in closed form (see e.g. Graybill, 1983).

Secondly, the lower bound $\min_i \theta > 1.5$ was chosen to allow us scale through the sum in (4.31) and, in practice, alternative conditions based on combinations of different θ_i may be more useful. For example, if θ_i is small for a single component of x and large for all other components, we might consider a bound which scales with $\tilde{p} - 1$.

Thirdly, Näther and Šimák (2003) perform a related calculation to ours in the univariate case ($p = 1$) where they compare the integrated resolved variance obtained by observing derivatives of order k and l in the case that $\theta \to \infty$. Letting L_k and L_l be the corresponding limits, they show that, for $l \geq k$,

$$\frac{L_k}{L_l} = 2^{l-k} \text{ for Gaussian covariance function} \tag{4.37}$$

$$\frac{L_k}{L_l} = \prod_{i=k+1}^{l} \left(2 + \frac{2i+1}{2(\nu - i)}\right) \text{ for Matérn covariance function} \tag{4.38}$$

What we see is that the Gaussian covariance model, on which our result is

based, represents conditions under which we can get the most out of derivatives. The result for the Matérn although only valid for $p = 1$, shows that if we compare observing derivatives of order l to function values ($k = 0$), that the ratio of function value to derivative decreases monotonically to 2 as the parameter $\nu \to \infty$ and, for ν small, the ratio is large so that derivatives are relatively much less informative. This means that, if our covariance function is misspecified as Gaussian, then the emulator model will resolve variance optimistically each time we observe derivatives. On the other hand, the ratio of derivatives to function values for our Gaussian result in (4.35) is independent of Θ (subject to its components being large enough for our approximations to hold) so that we can choose a value of Θ conservatively and still maintain this ratio. In any case, we believe that the infinite differentiability of the Gaussian covariance function is appropriate for the simulator we consider, as we discuss in Section 4.6.

Finally, we could consider higher order interactions between design points. For example, a natural extension might be to consider the trade-off between function values and derivatives for a set of m independent input points and n pairs of points which interact with each other but are independent of all other points.

4.5 Estimating Θ covariance parameters

Estimating components of Θ for the ϵ-surface covariance kernel is a difficult problem. One approach is to use maximum likelihood estimation as in Ripley (1988). The maximum likelihood approach relies on distributional assumptions, which we are keen to avoid for reasons discussed in Chapter 3, and is often beset by computational problems, but it will be useful to use it for comparison with other estimates and also to demonstrate the essence of the usefulness of derivatives in estimating Θ values. (For an alternative justification of this likelihood, which does not rely on distributional assumptions, a *quasi-likelihood* approach can be used. See e.g. Pawitan, 2001) Suppose that our emulator model is

$$s(x) = g(x)^T\beta + \epsilon(x) \tag{4.39}$$

where $\epsilon \sim \text{MVN}(0, K)$ (note we have dropped the δ-surface as its inclusion is problematic). Now we derive the maximum likelihood estimate (MLE) for (β, ϕ) where ϕ is the set of hyper-parameters that appear in K. The log likelihood is

$$L(\beta, \phi|S) = \text{const} - \tfrac{1}{2}\left(\ln|K_\phi| + (S - G\beta)^T K_\phi^{-1}(S - G\beta)\right) \tag{4.40}$$

Discarding the constant, as we may, the MLE of β minimises the quadratic form, and so is the generalised least squares estimate $\hat{\beta}$ given by

$$L^{-1}S = L^{-1}G\hat{\beta} \tag{4.41}$$

where L is lower triangular and defined by the Cholesky decomposition, LL^T, of K. Thus the profile likelihood, L_P, for ϕ is

$$L_P(\phi|S) = -\tfrac{1}{2}\left(\ln|K_\phi| + (S - G\hat{\beta})^T K_\phi^{-1}(S - G\hat{\beta})\right) \tag{4.42}$$

Writing $K = \sigma^2 R$, we can extract an MLE for σ^2

$$L_P(\phi) = -\tfrac{1}{2}\left(n\ln|\sigma^2| + \ln|R_\phi| + \frac{1}{\sigma^2}(S - G\hat{\beta})^T K_\phi^{-1}(S - G\hat{\beta})\right) \tag{4.43}$$

and hence we can extract an MLE, $\hat{\sigma}^2$, for σ^2

$$\hat{\sigma}^2 = \frac{1}{n}(S - G\hat{\beta})^T R_\phi^{-1}(S - G\hat{\beta}) \tag{4.44}$$

If $\phi = (\sigma^2, \theta)$, then

$$L_P(\theta|Z) = -\tfrac{1}{2}(n\ln\hat{\sigma}^2(\theta) + \ln|R_\theta| + n) \tag{4.45}$$

The attraction of the MLE is that we can easily include derivatives into the analysis by redefining $s(x)$ to be the vector of itself and the derivatives of interest, and similarly for $g(x)$ and $\epsilon(x)$. Then G is the matrix of g and its derivatives at X, S is the vector of observed function values and derivatives, and covariances involving derivatives of $\epsilon(x)$ are simply absorbed into K_ϕ with the scale parameter, σ^2, extractable as before. Figure 4.2 shows us the potential derivatives have in offering us information about Θ. The Figure shows the profile likelihood for θ for ten samples from a univariate process, $\epsilon(x)$, with $\text{Cov}[\epsilon, \epsilon'] = e^{-7.5(x-x')^2}$, with each sample of observations at seven input points. The top plot shows the profile

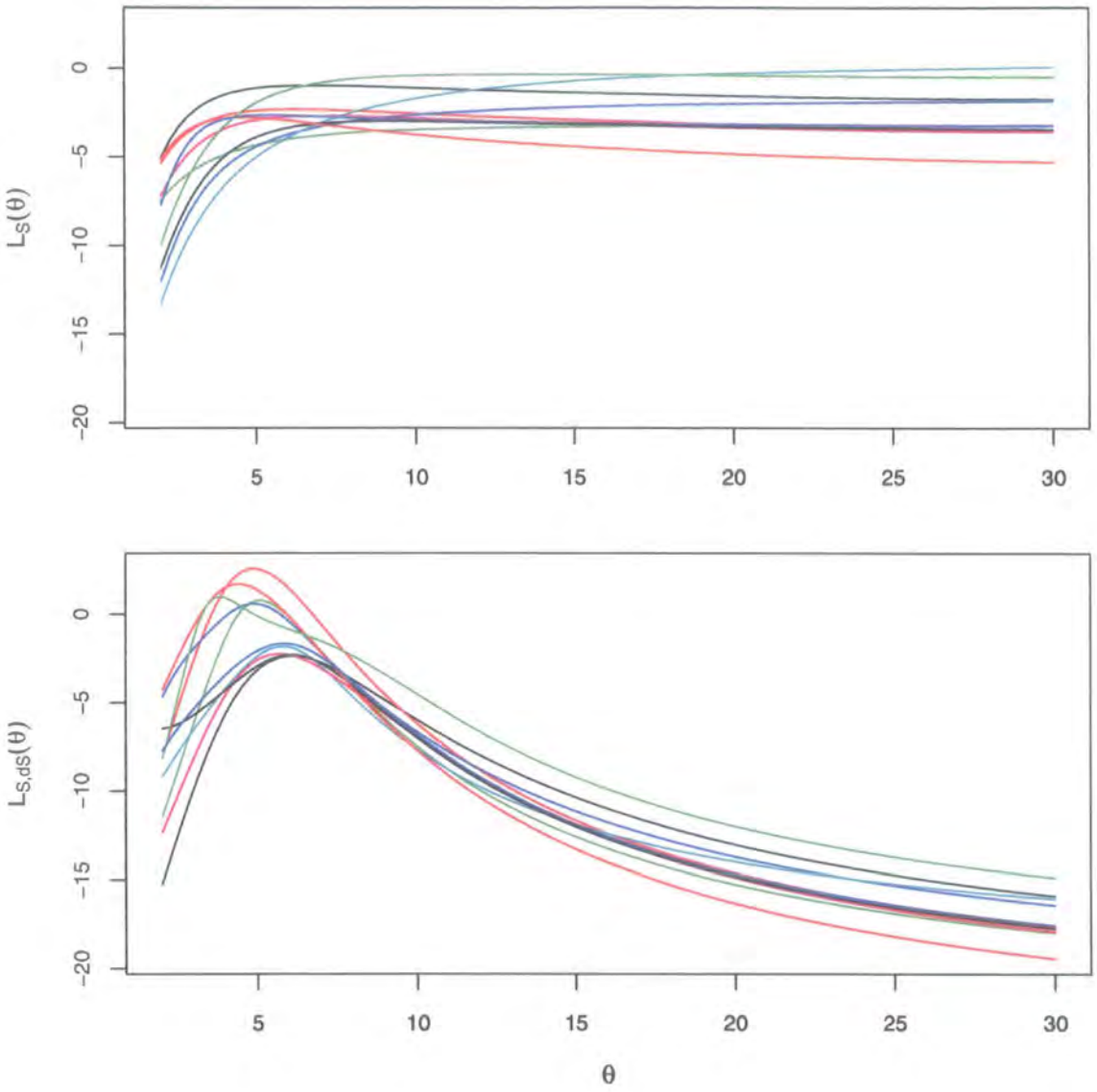


Figure 4.2: Profile likelihood for θ for ten samples from a Gaussian Process with $r(h) = \exp^{-7.5h^2}$, each sample taken at seven points, without derivatives (top) and with derivatives (bottom).

likelihood based on observing function values only and the bottom plot shows the profile likelihood when derivative also are observed.

One of the problems of maximum likelihood estimation has been that a sparsely scattered collection of function values simply does not offer enough information to distinguish between competing θ values. The top plot in Figure 4.2 shows this with the profile flat for all but very small θ values. Derivatives look as though they will

solve one of the problems of MLE estimation; the information about Θ is there waiting to be extracted. MLE estimation is still problematic though. Firstly, it relies on full distributional assumptions that we can not justify and are thus keen to avoid. Secondly, it is computationally intensive, particularly in higher x dimensions and when we do not wish to set all components of Θ to be equal. What we want is a computationally cheaper way of extracting information contained in derivatives which does not rely on full distributional assumptions. In Chapter 6, we develop such an estimate and apply it to our problem.

4.6 Choice of Gaussian covariance function

Throughout the thesis, we work almost exclusively with the Gaussian covariance function. This form has come in for criticism recently: Stein (1999), for example, is highly critical of using the function to model physical processes in the context of interpolation of spatial data, showing that it can lead to predictions that are much too optimistic in resolving uncertainty when the covariance function is misspecified and advocating a Matérn form. It is better to be badly conservative than wildly optimistic, Stein argues. Whilst agreeing with his sentiments, we do not necessarily feel that they transfer to the context of computer simulators. Put another way, we agree that there is often no basis for knowing *a priori* the degree of smoothness of some physical process; but we have much stronger beliefs concerning the smoothness of computer simulators of the type discussed in this thesis. It is true that an optimistic misspecification of Θ is potentially more dangerous with the Gaussian than Matérn correlation function, but we have shown that derivatives offer the potential to provide good estimates of Θ values. Perhaps the one assumption most needing to be tested in both forms is that of a constant Θ across all the input space although, when this is not the case, a conservative Θ value towards the upper end of the range should suffice. Stein's comments do, however, focus the mind on the need for careful diagnostics relating to such specifications as mentioned at the end of Section 4.3. In our analysis of the plankton problem in Chapter 6, we introduce and apply diagnostics to back up our belief statements and, in doing so, demonstrate another

area in which derivative information can be useful.

Chapter 5

Taking derivatives in compartmental simulators

In the previous chapter, we showed that the framework from the computer simulator literature could be naturally extended to allow for inclusion of derivatives in emulator building and refinement. In this chapter, we develop a cheap and efficient method for generating such derivatives in the case of a compartmental simulator. We also perform calculations to estimate the increase in cost, in CPU run-time, of computing derivatives so that, by weighing the increase against the additional variance resolved, we can ascertain whether it is worth our while computing a given collection of derivatives. The model in Chapter 2 is used to clarify aspects of the approach, although the method itself is perfectly general to the class of compartmental simulators.

5.1 Derivatives and compartmental structure

The calculation of derivatives has long been central to the field of *sensitivity analysis*, in which so-called perturbation methods, in which 'pseudo' derivatives are generated by perturbing the inputs and re-running the simulator at the new values, have proved popular. The advantage is that the method can be applied to any simulator; however Oblow et al. (1986), who give a brief overview of the field, point to several difficulties: "the time-consuming nature of the perturbation calculations and the

inaccuracies that occur in trying to estimate sensitivities from small changes in inherently imprecise numbers." Both of these difficulties are of particular concern to us. The precision issue is important because we are concerned with actual derivatives rather than "effects of small changes" and we are loathe to model our derivatives as being observed with error. Moreover, we may be interested in observing higher order derivatives as well, for which perturbation methods become yet less reliable. The time issue is equally as important: using perturbation methods, even the lowest order approximation requires an extra run per each extra derivative to be computed. Since the simulators we consider are generally high dimensional in input space, we require a much more efficient method if the use of derivatives is to be practical.

Here we chose to differentiate the equations by hand, which is both practical for a model the size of ours and avoids this initial overhead. Our ability to do this successfully and cheaply in the case of a compartmental model lies in recognising our *simulator* as two distinct parts.

Firstly, the compartmental model itself defines the following *initial value problem*

$$\frac{dy}{dt}(t,x) = f(t,y,x),\ y(t_0,x) = y_0,\ f : R \times R^k \times R^p \rightarrow R^k, \tag{5.1}$$

where $k = \dim(y)$, the number of compartments, and $p = \dim(x)$, the number of input parameters. For given x, there exists a unique solution $y(t,x)$ to (5.1) provided $f(t,y,x)$ satisfies a Lipschitz condition, satisfied automatically when f is continuously differentiable in y for all (t,y) (See for example Burden and Faires, 1989).

Secondly, the simulator combines the initial value problem (5.1) with a solver routine which, for given x, produces an approximate numerical solution, $\tilde{y}(t,x)$, for $y(t,x)$. For the code provided for the model in Chapter 2, a fifth-order Runge-Kutta method was used (See Press et al., 1992). Runge-Kutta methods are a popular and often-applied general class of techniques for solving systems of ODEs. In the following section, we introduce a simple first-order Runge-Kutta method, Euler's Rule, and demonstrate our approach for obtaining derivatives. We also discuss computational savings made through exploiting compartmental structure.

5.2 Differentiating Euler's Rule

Consider the general system of ODEs in (5.1). For sufficiently small h,

$$\frac{dy}{dt}(t,x) \approx \frac{y(t+h,x) - y(t,x)}{h}$$

Substituting this into (5.1) and rearranging, we have

$$y(t+h,x) \approx y(t,x) + hf(t, y(t,x), x)$$

Writing $y_n = y(t_n, x)$, we obtain *Euler's Rule*

$$\tilde{y}_{n+1} = \tilde{y}_n + hf(t_n, \tilde{y}_n, x) \tag{5.2}$$

Fixing x, we can propagate (5.2) forward through time and obtain the solution, $\tilde{y}(t)$, *for this* x. However, viewing f and $\tilde{y}_n$ as functions of x then, provided f is differentiable in components of x to the required order, we can differentiate both sides of (5.2) and obtain similar expressions which relate derivatives of $\tilde{y}_{n+1}$ to those of $\tilde{y}_n$. For given f, we can use a computer algebra package to generate the algebraic relations once, add these expressions into the simulator code and let the simulator propagate the numerical expressions forward through time in exactly the same way as for function values. In particular, we can alter f in the computer code to additionally return its derivatives at each iteration and simply extend the vector y_n in (5.2) to include the corresponding derivatives.

5.3 Higher order generalisation and adaptive step-size

Euler's Rule is of conceptual importance since the higher order methods which are often used in practice all emanate from it. The general s-stage Runge-Kutta method is given by

$$y_{n+1} = y_n + h\sum_{i=1}^{s} c_i k_i, \text{ where } k_i = f\Big(t_n + a_i h,\ y_n + h\sum_{j=1}^{i-1} b_{ij}k_j,\ x\Big) \tag{5.3}$$

where the a_i, b_{ij} and c_i are constants chosen to satisfy certain error properties. From (5.3), we see that each step of the general s-stage method is a linear combination

of s Euler-style steps. Hence we can simply redefine the vector of functions, f, to include derivatives by altering f to additionally return these derivatives in the code. All this then requires is the dimension of the vector y_n to be increased and the remainder of the code remains unaltered.

Often, as was the case with the code provided for our model, a Runge-Kutta solver comes with an adaptive stepsize routine, which aims to achieve a specified accuracy in the solution whilst minimising computational load. It does so by estimating a vector, Δ, of truncation errors - one for each output - at each iteration and adjusting the step-size accordingly; if the errors are much smaller than required, the step size for the next iteration is increased and, if the errors are larger than allowed for the required accuracy, the step-size is decreased and the iteration repeated. Typically, this adjustment is based on $\max(\Delta)$, so that we guarantee a specified level of accuracy for all outputs (See Press et al., 1992, for more details).

As we have already remarked, inclusion of derivatives in the Runge-Kutta updating is handled in the same way as the original code, by a simple extension of the vector y_n in (5.3) to also include derivatives. This means that truncation error estimates are naturally generated by the code for derivative quantities, which in turn presents us with something of a dilemma. For any iteration, Δ contains all the original truncation errors relating to function values plus additional estimates relating to derivatives. If we adjust the stepsize based on $\max(\Delta)$ then, for any given iteration, this quantity must be at least as large as it would have been for the original Δ (since all we have done is added some additional estimates relating to derivatives). The issue here is that if we are truly 'differentiating the simulator', then this differentiation should not change the simulator; in other words, we want the derivatives of the output at the time points which would have been given by the original code.

At the same time, maximising over derivative error estimates in the adaptive routine, offers a somewhat tempting diagnostic on the quality of the simulator, with a large increase in the number of steps required suggesting a mismatch between derivatives of the simulator and derivatives of the differential equations. We expect some increase since including derivatives in the adaptive routine increases the di-

mension of the vector Δ over which the maximisation takes place so that doing so can only affect the stepsize by decreasing its value.

In practice, the nature of the criterion in adjusting stepsize based on the maximum error together with tight error bounds (our code requires accuracy of the solution to 1 in 10^5 for each component of y_n) means that any change in stepsize caused by derivatives will change function values by a negligible amount for our simulator so that little should be lost by restricting the truncation error assessment to the original output quantities only.

Since our model was cheap to run, we choose to include first-order derivatives for the thirteen inputs in Table 2.1 in the adaptive routine, which led to an average increase of 14.4% in the number of steps required over twenty five runs, with the smallest increase 4.6% and the largest 44.6%. We considered these increases not to represent any cause for concern given the nature of the adaptive routine. When considering additional cost of derivatives, however, we do not count this additional cost in the comparison since it represents extra accuracy in relation to the differential equations which is not strictly required.

5.4 Exploiting compartmental structure

For the process of derivative computation described in Section 5.2, computational savings arise in two ways; firstly, through the additive structure of compartmental models in general and, secondly, by consideration of the degree of "localisation" of input parameters in a given model, where the notion of "localisation" is one we will make explicit shortly.

By the additive structure of compartmental models, we mean simply that the time derivative of any compartment is given by the sum of functions flowing into the compartment minus the sum of functions flowing out. Hence, writing $\Lambda = (\Lambda_1, \Lambda_2)^T$ for the set of r flow functions in a given k-compartment model, where Λ_1 is the set of r_1 flow functions that are connected to a single compartment only and Λ_2 is the set of r_2 flow functions which link two compartments together, we can write

$$f = (I_1|I_2)\Lambda \tag{5.4}$$

where I_1 is a $k \times r_1$ diagonal matrix with ± 1 in its diagonal entries and I_2 is a $k \times r_2$ anti-symmetric matrix with zeros on the centre diagonal and ± 1 on the two diagonals either side. For the model described in Chapter 2, we have $\Lambda_1 = (\Lambda_P, \Lambda_Z, \Lambda_N)$ and $\Lambda_2 = (\Lambda_{ZN}, \Lambda_{NP}, \Lambda_{PZ})$. Then, writing (5.4) explicitly for the model in Chapter 2,

$$f = \left(\begin{array}{ccc|ccc} -1 & 0 & 0 & 0 & 1 & -1 \\ 0 & -1 & 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 1 & -1 & 0 \end{array}\right) (\Lambda_P, \Lambda_Z, \Lambda_N, \Lambda_{ZN}, \Lambda_{NP}, \Lambda_{PZ})^T \tag{5.5}$$

For $x = (x_1, \ldots, x_p)$, we can apply $\nabla_x = (\partial/\partial x_1, \ldots, \partial/\partial x_p)$ to (5.4) and obtain

$$\nabla_x f = (I_1 | I_2) \nabla_x \Lambda \tag{5.6}$$

where, on the left hand side, $\nabla_x f$ is a $k \times p$ matrix of functions and, on the right hand side, ∇ acts on each arc-function in Λ, producing a p-dimensional row vector for each. Note that we can readily replace x by any subset of x in (5.6) and hence we can evaluate as few or as many input derivatives as we wish. If, for example, some x-derivatives are difficult to derive or computationally relatively expensive to evaluate, they can be dropped.

For example, consider our model with ∇_x operating on

$$\Lambda_P = \phi_P P + \tfrac{m+h^+}{M} P,$$

the flow function describing exports from the phytoplankton compartment. In doing so, we first split Λ_P down into two additive parts,

$$\Lambda_P^1 = \phi_P P, \quad \Lambda_P^2 = \tfrac{m+h^+}{M} P$$

Then taking each term to be a generic function of x and applying ∇_x, we obtain

$$\nabla_x \Lambda_P^1 = \phi_P \nabla_x P + P \nabla_x \phi_P \tag{5.7}$$

$$\nabla_x \Lambda_P^2 = \frac{m + h^+}{M} \nabla_x P + \frac{P}{M} \nabla_x m \tag{5.8}$$

Eqns. (5.7) and (5.8) demonstrate the essence of our approach: the expressions are easily generated, by hand or using a computer algebra package, and are linear in the derivative vectors - in this case $\nabla_x P$ - so that derivatives with respect to many inputs can be generated in the code quickly using efficient vector operation

implementations. The vector argument $\nabla_x P$ is then propagated forward, using (5.3), in the same way as P. The vectors $\nabla_x \phi_P$ and $\nabla_x m$ are fixed constant vectors with jth component defined by $(\nabla_x x_i)_j = 1$ if $i = j$ and zero otherwise (where in our case ϕ_P corresponds to x_1 and m to x_{13}). They constitute extra terms generated by taking derivatives with respect to inputs which appear explicitly in the expression for Λ_P. The fewer of these extra terms there are, the more we will feel the benefit of the vector implementation and the cheaper derivatives will be relative to function values. Hence, if most additive terms have explicit dependence only a small subset of x, then the relative cost of computing derivatives will remain relatively cheap; an idea which we make more concrete in Section 5.6. Firstly, however, we show how to implement the calculations required to obtain (5.7) and (5.8) using Maple.

5.5 Implementation in Maple and R

In this section we demonstrate how to implement the example in (5.7) and (5.8) in Maple and then how to add to the simulator code. In both sets of code, Λ_P^1 is denoted `Pdeep` and Λ_P^2 is denoted `Pflux`. Any input we enter into Maple begins on a new line with the prompt `>` and is executed with the `;` command. For each input line, the subsequent Maple output is shown on the following line.

Starting with $\Lambda_P^1 = \phi_P P$, we define each term to be a generic function of x:

```
> Pdeep := phiP(x)*P(x);
                          Pdeep := phiP(x) P(x)
```

Then we use the `diff` command to differentiate with respect to x:

```
> dPdeep := diff(Pdeep, x);
                    / d          \                 / d       \
          dPdeep := | -- phiP(x)| P(x) + phiP(x) | -- P(x)|
                    \ dx         /                 \ dx      /
```

The process is similar for $\Lambda_P^2 = \frac{m+h^+}{M} P$, although now the mixed layer depth M and h^+ forcing functions are independent of x:

```
> Pflux := ( (m(x) + hplus)/M )*P(x);

                                  (m(x) + hplus) P(x)
                       Pflux := -------------------
                                          M
```

Applying the `diff` command,

```
> dPflux := diff(Pflux, x);

                       / d      \                       / d      \
                       | -- m(x)| P(x)   (m(x) + hplus) | -- P(x)|
                       \ dx     /                       \ dx     /
            dPflux := -------------- + ------------------------
                            M                     M
```

The results on the Maple working can then be added into the simulator code (given here in R syntax) as follows:

```
Pdeep <- x[1]*P
dPdeep <- x[1]*dPdx
dPdeep[1] <- dPdeep[1] + P
```

The first line calculates `Pdeep`, as would be required in the original simulator without derivatives, where `x[1]` corresponds to ϕ_P. The second line computes the vector of derivatives, `dPdeep`, in terms of the vector, `dPdx`, where both these vectors are of the same dimension as the vector, `x`, of input values. The final line adds the extra contribution to the first component of `dPdeep` caused by the explicit appearance of `x[1]` in `Pdeep`.

Similarly for Λ_P^2:

```
Pflux.denom <- x[13] + hplus
Pflux <- ( Pflux.denom / M ) * P
dPflux <- Pflux.denom * dPdx
dPflux[13] <- dPflux[13] + P
dPflux < - dPflux/M
```

One thing to note is that computing derivatives will require additional storage space. In particular, as well as the final derivative values, note that in the R code, we have introduced a temporary variable, `Pflux.denom`, which we would not have defined in the original code. The reason for this is to save us duplicating the same (very cheap) calculation twice. This is simply efficient coding and does not cause any problem, but it is perhaps worth remarking here that, for larger models, it is likely that we will generate a significant number of additional temporary variables when we include derivatives calculations so that extra storage space is required by the code. In general, we would not expect these extra storage requirements to cause a problem and, overall, there may be no net increase in storage requirements if calculating derivatives allows us to perform less runs on the simulator for the same information.

5.6 Additional cost of derivatives: a heuristic

In this section we develop a heuristic for the proportional increase in cost (in computer run-time), $\pi_{\tilde{x}}$, in order to calculate first order derivatives with respect to a given subset $\tilde{x}$ of x (for example, we might take $\tilde{x}$ to be the set of active inputs x^*).

5.6.1 Derivation of heuristic

Let $\Lambda(x^*, y^*(x))$ be a scalar function of vectors $x^* \subseteq x$ and $y^* \subseteq y$. Then, applying the chain rule for differentiation,

$$\nabla_x \Lambda(x^*, y^*(x)) = \nabla_{x^*} \Lambda(x^*) + \sum_{y_i \in y^*} \frac{\partial \Lambda(y^*)}{\partial y_i} \nabla_x y_i \tag{5.9}$$

Let $\dim(x) = p$, $\dim(x^*) = p^*$, $\dim(y) = k$, $\dim(y^*) = k^*$ denote the cost, in some standardised units, of multiplying a scalar by a vector, dimension d, by M_d and the cost of adding two vectors, both dimension d, by A_d. Since we are interested in comparing the cost of function values and derivatives, we write the cost of evaluating Λ as the sum of two parts, $\text{Cost}(\Lambda) = C + K$, where K is the 'set-up' cost of Λ, constituting calculations required to evaluate Λ which can be re-used when calculating derivatives and C is the cost of the remaining calculations

required to evaluate Λ. Next we assume that each $\frac{\partial \Lambda(y^*)}{\partial y_i}$ and each component of $\nabla_{x^*}\Lambda(x^*)$ costs C to evaluate (we discuss this assumption at the end of this section) so that, counting operations in (5.9), we find

$$\begin{aligned} \text{Cost}(\nabla_x \Lambda(x^*, y^*(x))) &= p^*C + p^*A_1 + k^*(C + M_p) + (k^* - 1)A_p \\ &= (p^* + k^*)C + (k^*)M_p + (k^* - 1)A_p + p^*A_1 \quad (5.10) \end{aligned}$$

Computational savings come from the fact that the dependency of M_d and A_d on d is typically weakly linear and, moreover for the values of d we consider, $M_d \approx M_1$, $A_d \approx A_1$. As multiplication is more expensive than addition, from (5.10) we see that if $C \gg M_1$, as typically it is for the functions we consider, then the additional cost of evaluating the p derivatives of Λ is approximately $(p^* + k^*)C$.

For a compartmental model with r flow functions, $\Lambda_1, \ldots \Lambda_r$, let K be the sum of the flow-function set-up costs and $C_1, \ldots, C_r$ the remaining costs. Then, a natural generalisation is to compare $K + \sum_{i=1}^r C_i$ with $K + \sum_{i=1}^r C_i(1 + k_i^* + p_i^*)$. Supposing that the costs, $C_1, \ldots, C_r$, are roughly the same (again we discuss this at the end of the section), then the proportional increase in cost, π_x, can be estimated by

$$\pi_x = 1 + (L_x + L_y)\rho, \text{ where } L_x = \frac{1}{r}\sum_{i=1}^r p_i^*, \; L_y = \frac{1}{r}\sum_{i=1}^r k_i^*, \; \rho = \frac{C}{C + K}.$$

Note that we can easily generalise this if we only wish to differentiate with respect to components of a subset $\tilde{x}$ of x by replacing x by $\tilde{x}$, which in turn requires us to replace p_i^* by $\tilde{p}_i^*$

$$\pi_{\tilde{x}} = 1 + (L_{\tilde{x}} + L_y)\rho, \text{ where } L_{\tilde{x}} = \frac{1}{r}\sum_{i=1}^r \tilde{p}_i^*, \; L_y = \frac{1}{r}\sum_{i=1}^r k_i^*, \; \rho = \frac{C}{C + K}. \quad (5.11)$$

In (5.11), K is the 'setup cost' for a run, representing calculations required to evaluate the r flow functions that can be reused in calculating derivatives, and C is the average cost, over flow functions, of operations not included as part of the set up cost K. The terms $\tilde{p}_i^*$ and k_i^* are the number of inputs in $\tilde{x}$ and the number of outputs, respectively, on which the ith flow function has explicit dependence in the model equations. Hence $L_{\tilde{x}}$ and L_y can be thought of as measuring the degree of localisation of $\tilde{x}$ and y respectively.

From (5.11), we see that the smaller the values of $L_{\tilde{x}}$, L_y, and ρ, the smaller the additional cost in generating derivatives. The term L_y, which is fixed for any

simulator, can be thought of as a setup cost for generating derivatives and hence the lower bound, π_x^L, for π_x is obtained by taking $L_x = 0$ in (5.11).

Note that $\pi_{\tilde{x}}$ only depends on $\tilde{x}$ through $L_{\tilde{x}}$; hence if we run the simulator, setting it to calculate derivatives with respect to inputs in $\tilde{x}$, and record the observed proportional increase in cost, $\pi_{\tilde{x}}$, we can substitute this and the value of $L_{\tilde{x}}$ into (5.11) and fix ρ. Having fixed ρ, we can calculate $L_{\tilde{x}'}$ for any other subset $\tilde{x}' \subseteq x$ and use (5.11) to calculate the corresponding $\pi_{\tilde{x}'}$.

5.6.2 Discussion of heuristic assumptions

In deriving, (5.11) we made two assumptions: (i) for an individual flow function, Λ, the non-reusable cost C was of the same order as the cost of calculating each component of $\nabla_{x^*}\Lambda(x^*)$ and (ii) that the C corresponding to different flow functions were all of the same order.

For assumption (i), consider the example of Λ_P at the end of Section (5.4). Counting the cost, K, of reusable operations and the costs of non-reusable operations, C, we obtain:

$$\Lambda_P^1 \ : \tag{5.12}$$

$$K = 0$$

$$C_{\Lambda_P^1} = M_1, \quad C_{d\Lambda_P^1} = M_d + A_1$$

$$\Lambda_P^2 \ : \tag{5.13}$$

$$K = \text{Cost}(M) + \text{Cost}(h^+) + A_1$$

$$C_{\Lambda_P^2} = M_1 + D_1$$

$$C_{d\Lambda_P^2} = M_d + D_d + A_1$$

where D_d is defined to be the cost of dividing a d-vector by a scalar (which is trivially equal to $M_1 + M_d$). From our earlier remarks that $M_d \approx M_1$ and $A_d \approx A_1$, the assumption seems to be valid, for Λ_P at least. Note also that, for Λ_P^1, the set-up cost, K, involves the evaluation of the mixed layer forcing function, M, and its time derivative dM, through $h^+ = \max(0, dM)$. These operations involve searching and interpolation of tables of forcing function values so that the reusable operations are

relatively expensive and hence including derivatives does not add much extra cost, proportionally, in this case. Assumption (ii) is more difficult to justify although we believe it to be reasonable magnitude of order approximation; in reality, we do not expect the assumption to be strictly true but rather that competing forces cancel out to make it a reasonable approximation. This appears to be borne out in Section 5.7 in comparisons of predicted proportional increases with those observed.

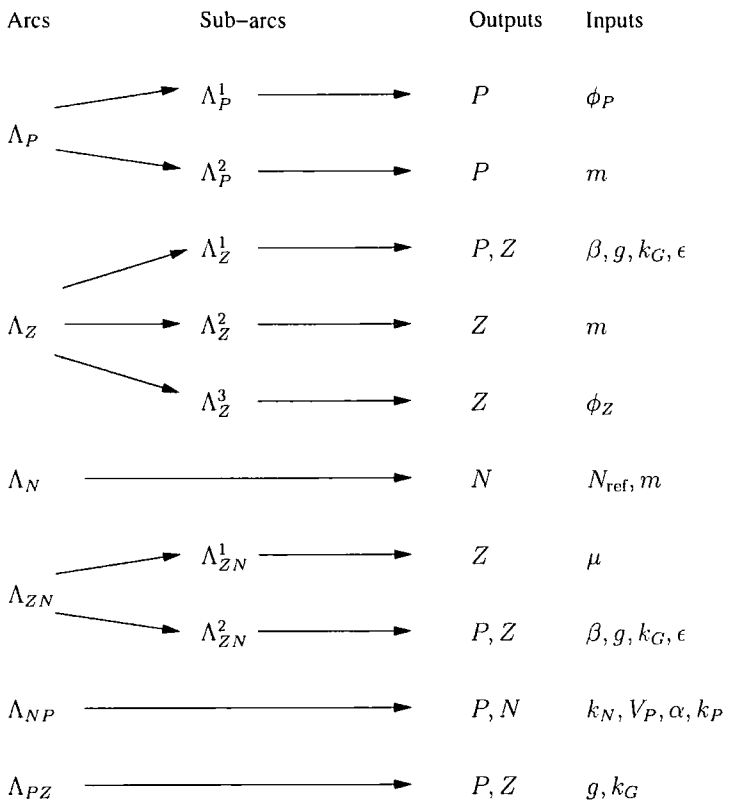
5.7 Localisation of inputs in the PZN model

As an example, consider the additional cost of generating derivatives with respect to all thirteen inputs in our model. Figure 5.1 lists the inputs and outputs appearing explicitly on the different flow functions of the PZN model, where we have split some of the arcs down further into their additive components, as we did with the example in (5.7) and (5.8). Splitting the arcs denoting flow functions into additive pieces is natural for two reasons. Firstly, linearity properties of derivatives allow us to add derivatives of two sub-arcs to obtain the derivative of the arc. Secondly, different additive pieces tend to correspond to different processes in the physical modelling which contain their own localised input parameters. For example, the two sub-arcs in (5.7) and (5.8) capture very different processes which combine to form phytoplankton exports: phytoplankton mortality modelled by Λ_P^1 and physical dilation as a result of vertical mixing processes, modelled by Λ_P^2.

From Figure 5.1 we see that, of the ten arcs, there are five which depend on one input, two which depend on two and three which depend on four so that $L_x = ((5 \times 1) + (2 \times 2) + (3 \times 4))/10 = 2.1$. Counting numbers of outputs, six arcs depend on one output and four on two outputs so that $L_y = ((6 \times 1) + (4 \times 2))/10 = 1.4$. Running the simulator at twenty five input values, the proportional increase in CPU time caused by generating the thirteen derivatives was 2.6. The proportional increase in individual runs varied from 2.1 to 3.3 with a standard deviation of 0.25. This variation is largely random and down to the effect of other processes on the computer so that 2.6 was thought be a good estimate of the proportional increase of any set of twenty five runs. Taking $\pi_x = 2.6$, and substituting into (5.11) together

with $L_x + L_y = 3.5$, we obtain the corresponding vector $\rho = 0.46$.

Figure 5.1: Input and outputs appearing on each additive arc of the model.



The reason for a lower value of C relative to K ($\rho = 0.46$ corresponds to $C = 0.85K$), and so smaller increase in cost incurred by calculating derivatives, is that the partial derivative calculations in the model re-use forcing function values calculated in evaluating the flow functions, and these operations are relatively expensive, requiring searching and interpolation of look-up tables. In general, we may only wish to compute derivatives of a subset $\tilde{x} \subseteq x$, for example the active inputs, which will reduce the additional cost further. For any subset, we can estimate the proportional increase in cost in doing so, using the value for ρ obtained above.

In our analysis of the problem in Chapter 6, the set of inputs $x^* = (x_1,\ x_4,\ x_6,\ x_{10},\ x_{13})$ turn out be active. From Figure 5.1, the input counts are (reading the sub-arcs from top to bottom) 1, 0, 1, 0, 0, 1, 0, 1, 2, 1 so that $L_{x^*} = 0.7$. Plugging in the values for ρ and L_y into the heuristic for the increased cost, $\pi_{\tilde{x}} = 1 + (L_{\tilde{x}} + L_y)\rho$, we find $\pi_{x^*} = 1 + (0.7 + 1.4) \times 0.46 = 2.0$. Hence, over the twenty five input points, we expect to obtain most of the information given by derivatives, which is contained

within derivatives of active inputs, for a factor increase in cost of 2.0. Running the simulator at the twenty five runs gave a corresponding proportional increase of 2.1 so that the estimate is a very slight underestimate. Hence, for a reasonably large collection of input points, we expect to be able to generate derivatives with respect to active inputs at an overall cost factor of around 2.1.

It is the localised nature of the parameters with respect to which we differentiate, as measured through L_x and L_y, and the relatively high set up cost for a run - both features common to many compartmental models - which offers us the potential for computational saving and so increases the appeal of observing derivatives. Concerning compartmental models in general, the method will scale well with the dimension of x and y provided this localisation remains intact, so that the greater the number of compartments and the greater the number of inputs, the greater we can expect our information to computation ratio to be.

Essentially this calculation is a 'mean' calculation, where the cost of the various function derivatives relative to the original functions is treated as unknown. We could take more care in counting the operations carefully (see Griewank, 1989, for some interesting work using known relative costs of standard mathematical functions to bound costs of derivatives relative to the original function). Alternatively, we could perform numerical experiments to count the relative costs of different derivatives. In practice, we feel our heuristic offers a quick and reasonably method to estimate proportional increased cost in generating different collections of derivatives and would be particularly useful for larger compartmental models with many compartments and/or parameters.

As a final comment, we note that programs exist which automatically differentiate computer code (see *inter alia* Oblow et al. (1986) and Dimitrios et al. (2003)). Such programs offer potential to generate derivatives for lengthy code for which a 'by hand' approach may become infeasible. However, we expect to be able to get to a reasonably large scale with our approach, with the benefits being a retention of the efficiency which is forfeited in the automatic process through inefficiencies in the additional code generated which can often be significant.

Chapter 6

Emulator construction and refinement for the plankton model

In this chapter, we are concerned with emulator construction and refinement, using the Bayes Linear approach outlined in Section 3.5. We firstly demonstrate some uses of derivatives in forming prior emulators and secondly investigate additional reduction in uncertainty in the emulator achieved by including derivative observations as well as function evaluations. We begin by considering an exploratory site, then considering a single site, forming a prior based on the exploratory analysis, and investigating in detail the additional effects of derivatives in updating at this single site. Finally, we move to multiple sites and build emulators using function and derivative evaluations.

6.1 Choice of stations to emulate

In emulating the simulator at different locations, we consider the emulator model given in Eqns. (3.1) and (3.6),

$$s_s(x) = B_s g_s(x_s^*) + \epsilon_s(x_s^*) + \delta_s(x \setminus x_s^*), \tag{6.1}$$

where index s denotes station s and where we consider outputs separately at each station so that (6.1) is implicitly indexed by output at each station. The variation over the index s is resolved by differences in forcing functions at the different locations and so, in theory, we could learn about this variation by running the simulator

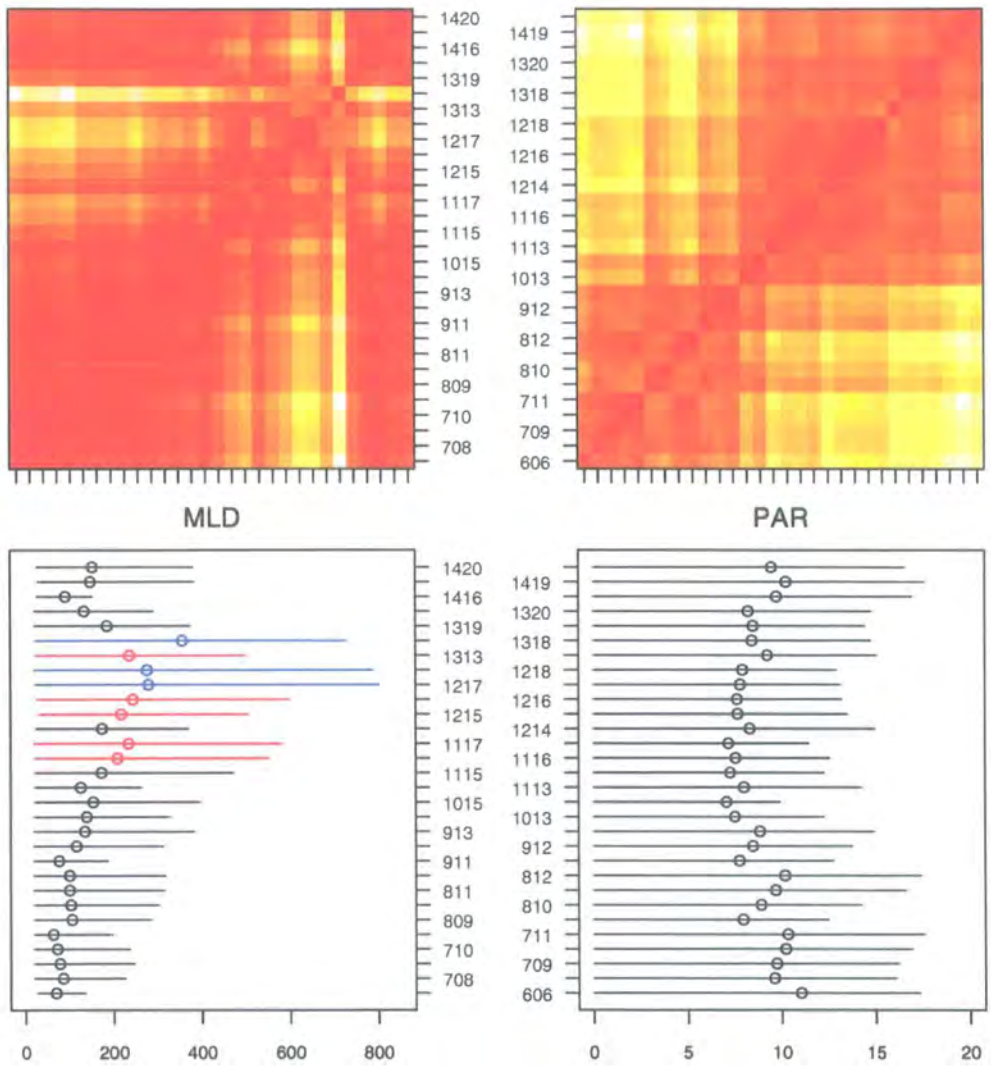


Figure 6.1: Top: Euclidean distances of mixed layer depth (MLD) forcing functions (left) and photosynthetically available radiation (PAR) forcing functions (right) between pairs of stations. Light colours correspond to large distances and dark colours to small distances. Bottom: Mean and range of forcing function values at each station, again with MLD shown left and PAR shown right; circles denote mean values and lines connect minimum and maximum values and, in the MLD plot, colours denote groupings suggested by the MLD distances.

at different locations for the same choice of input vector x. Hence the greater the 'difference' between forcing functions at two locations s and s', the less correlated we expect the quantities in the corresponding emulator models to be (for example, corresponding components of B_s and $B_{s'}$ in the case that $x_s^* = x_{s'}^*$ and $g_s = g_{s'}$). In

theory, this leads naturally into a multivariate covariance specification of unknown quantities across outputs and locations. However, in practice our beliefs do not stretch much beyond this initial statement since the model response is non-linear over time and the relationship between forcing functions, themselves time series, and simulator output is far from clear and a multivariate approach is difficult (though not necessarily impossible) to justify. The approach here then is to build emulators at individual sites and link these sites together by informal judgements, proceeding sequentially by using what we have learnt at previous sites to construct prior emulators at a given site and then updating the corresponding emulator formally based on simulator runs at this site. A consequence of this approach is that considering the sites in a different order would probably lead to different prior specifications at the sites; in practice, the many shared similarities of sites means that we would expect any differences caused by ordering to be relatively small. An alternative approach might be to use an exchangeability-type argument such as that made by Craig et al. (2001) to link beliefs about a fast and slow version of a simulator.

Figure 6.1, top row, shows the relative Euclidean distance between vectors of daily forcing functions values for all possible pair of stations, with mixed layer depth (MLD) shown left and photosynthetically available radiation (PAR) shown right. The bottom row shows the mean (denoted by a circle) and range of the forcing functions (denoted by the horizontal line from the minimum to maximum value) for each station. For MLD, the subset of three stations (1217, 1218, 1318) stand out as being far away from other stations, with station 1318 in particular registering large distances with other stations. Of the remainder, a subset of four stations - $(1115, 1116, 1215, 1216)$ - form a group within which all stations appear to be relatively close and outside of which stations are generally far apart. The remaining stations are all relatively close, generally increasing in distance with an increase in distance of their physical locations. For PAR there are two distinct groupings, the first consisting of stations up to and including station 913 and the second consisting of the remainder.

For simplicity, we aim to perform an emulation and subsequent calibration over a subset of the thirty stations and, based on Figure 6.1, we chose the subset

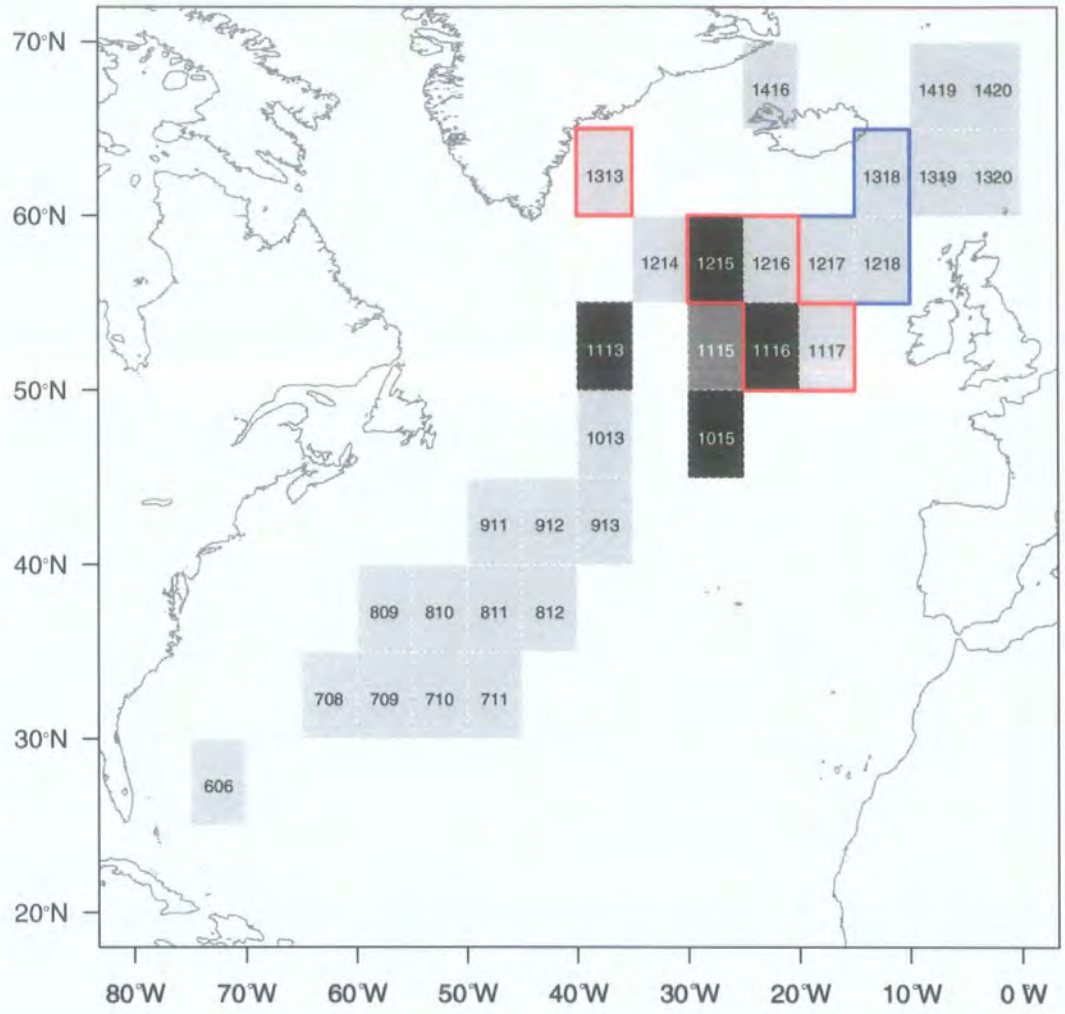


Figure 6.2: Map of stations shown with coloured outlines corresponding to the groupings identified by the mixed layer depth distances in Figure 6.1. The four stations marked black are those which we propose to emulate in this Chapter and then calibrate in Chapter 7. Station 1115, marked dark grey, is used to perform an exploratory analysis before we begin the emulation process.

(1015, 1113, 1116, 1215) along with station 1115 to perform an exploratory analysis. Figure 6.2 shows the stations, together with the groupings identified from the MLD distance analysis. These stations were chosen because they offered the chance to produce a calibration that was valid over a connected geographic region and, over which, stations covered a large part of the overall range of forcing functions so as to stretch the methodology. We see from Figure 6.2 that two of the four stations lie

in the main group, as determined by MLD distances in Figure 6.1, and two within the red region. Differences in MLD were the main consideration when looking for a range of physical conditions since the expert believed the MLD forcing function played a much larger role than PAR in determining simulator output.

6.2 Parameterisation of model output

The simulator output, for each station, is a time-series of the three populations at uneven time intervals. For nutrient, we have a single winter-time nitrate observation to calibrate against and we take this to correspond to simulator nutrient output, N_w, at the time, t_w, of the maximum MLD forcing function value at the station; this is the time at which nutrient is thought to correspond most closely to nitrate as discussed in Chapter 2. For phytoplankton, we have a time-series of physical observations and the question arises as to how best to parameterise this output. Our parameterisation is driven by the physical problem through our consideration of the main features of the data which we wish our calibrated model to reproduce; for example, the sudden increases or 'blooms' of phytoplankton in the springtime witnessed at the four stations and the general decline in phytoplankton stocks later in the year.

The strategy we adopt in attempting to pick up these features is by first interpolating simulator output at each station onto the time points of the corresponding physical observations. We then split the time series at a given station, s, into intervals according to the physical data and parameterise the phytoplankton output by the set $\bar{P}_s = (\bar{P}_{s,1}, \ldots, \bar{P}_{s,I_s})$, where $\bar{P}_{s,i}$ is the mean phytoplankton value in interval i, and I_s the total number of intervals, at station s. The resulting parameterisation for the four stations we consider is shown in Figure 6.3. As the mean is a linear operator, the derivative of an interval mean is simply the mean of the derivatives in the interval and so we can easily re-parametrise the derivatives generated by our simulator.

For phytoplankton blooms, choosing an interval about the maximum value and calibrating to the mean taken over the interval reflects our belief that changing inputs

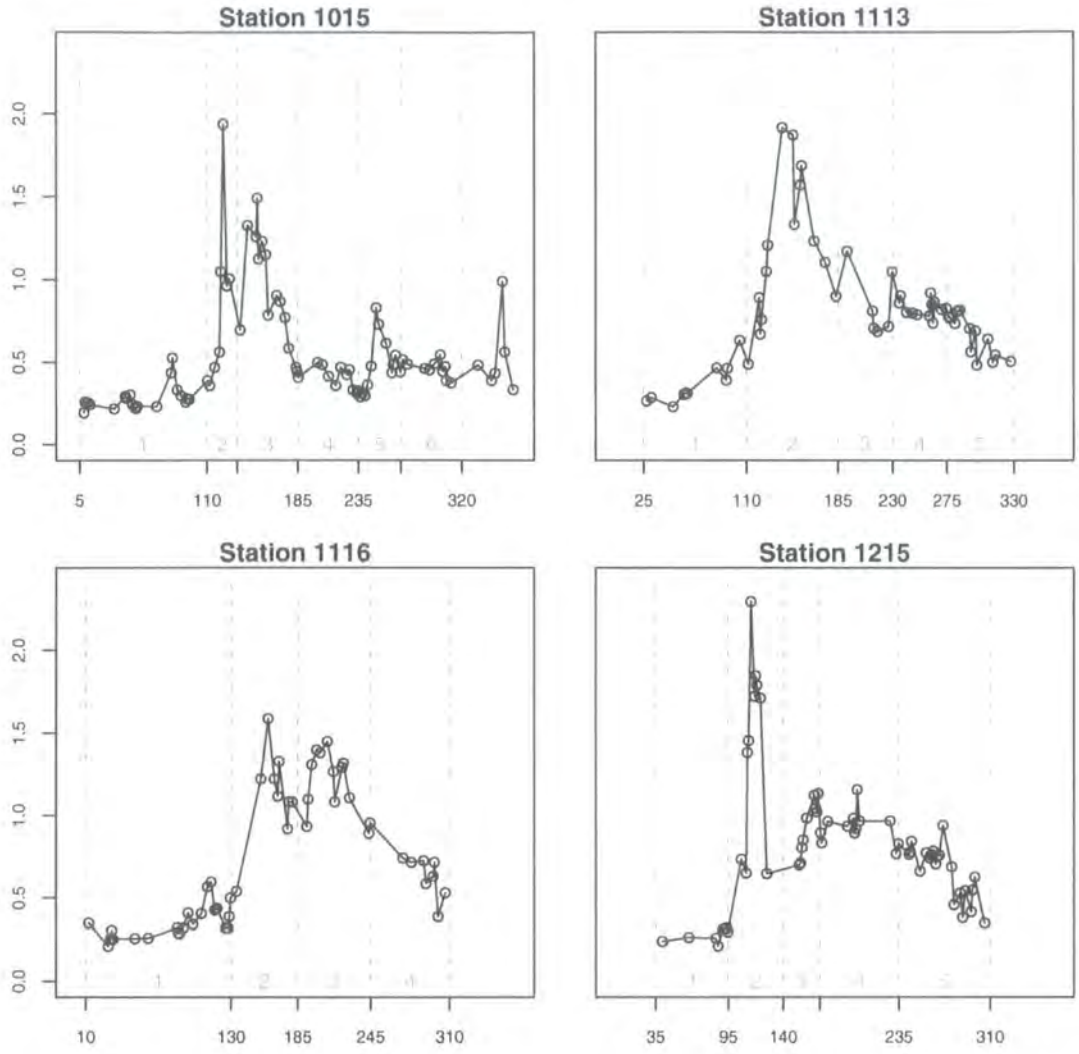


Figure 6.3: Observed chlorophyll (mmol m^{-3}) at the four calibration locations recorded for 1998 where $t = 0$ corresponds to midnight GMT on 31st December 1997. Vertical lines determine intervals used for parameterisation as described in Section 6.2.

can also affect the phase of the simulator bloom as well as its magnitude. Such a parameterisation is then appealing for two reasons: firstly, it yields a relationship between input and the (re-parameterised) output which is smoother than that based on output at selected time points and, secondly, in calibrating the model in Chapter 7, it allows us to score as a good match output with a similarly shaped but slightly-out-of phase bloom (calibration based on selected time points would lead to big miss-match scores for outputs either side of the bloom in such a situation.) We choose the end points of the interval to reflect our desire to judge an input as giving

a good match if it produces a bloom of roughly the correct shape anywhere within the interval. Calibrating to the mean levels in intervals either side of the bloom imposes restrictions that the raised mean level in the sandwiched interval corresponds to a reasonably isolated peak. We ignore what we observe to be a bloom at the end of the cycle at Station 1015 as there in no data to the right of the bloom for us to include in an interval around the bloom. Mean values over non-bloom intervals are also taken to capture the general level of phytoplankton at various times of year and are preferable to single time points because of the high day to day variability evident in the data. It allows us to capture, for example, the general pattern of decline from summer to autumn shown in stations 1113, 1116 and 1215.

It is worth noting that the fairly coarse representation of the output that this transformation offers us is quite desirable at this stage; we are looking for a parameterisation which allows us to rule out a large part of the space of possible inputs in our calibration. A simple re-parametrisation of outputs at the first stage of calibration is a good strategy in general because the parameterisation is not only applied to the physical data but to the simulator output and many simulators produce output over much of the input space which is very different to the physical data to which the simulator is to be matched.

6.3 Preliminary analysis at station 1115

There are many ways to construct emulators (See, for example, Oakley (2002) and Craig et al. (1998)). Our aim in this section is to demonstrate the role that derivatives can play in constructing the emulator. Generating derivatives brings additional information at a relatively cheap cost and, in addition, derivatives offer a somewhat different type of (localised) information. Hence our goals are, firstly, to use this extra information to improve our prior specification in general terms and, secondly, to divide up resources so that information is targeted at parts of the prior specification about which it is most informative.

In all that follows, x_i is used to denote the ith input in Table 2.1, linearly rescaled

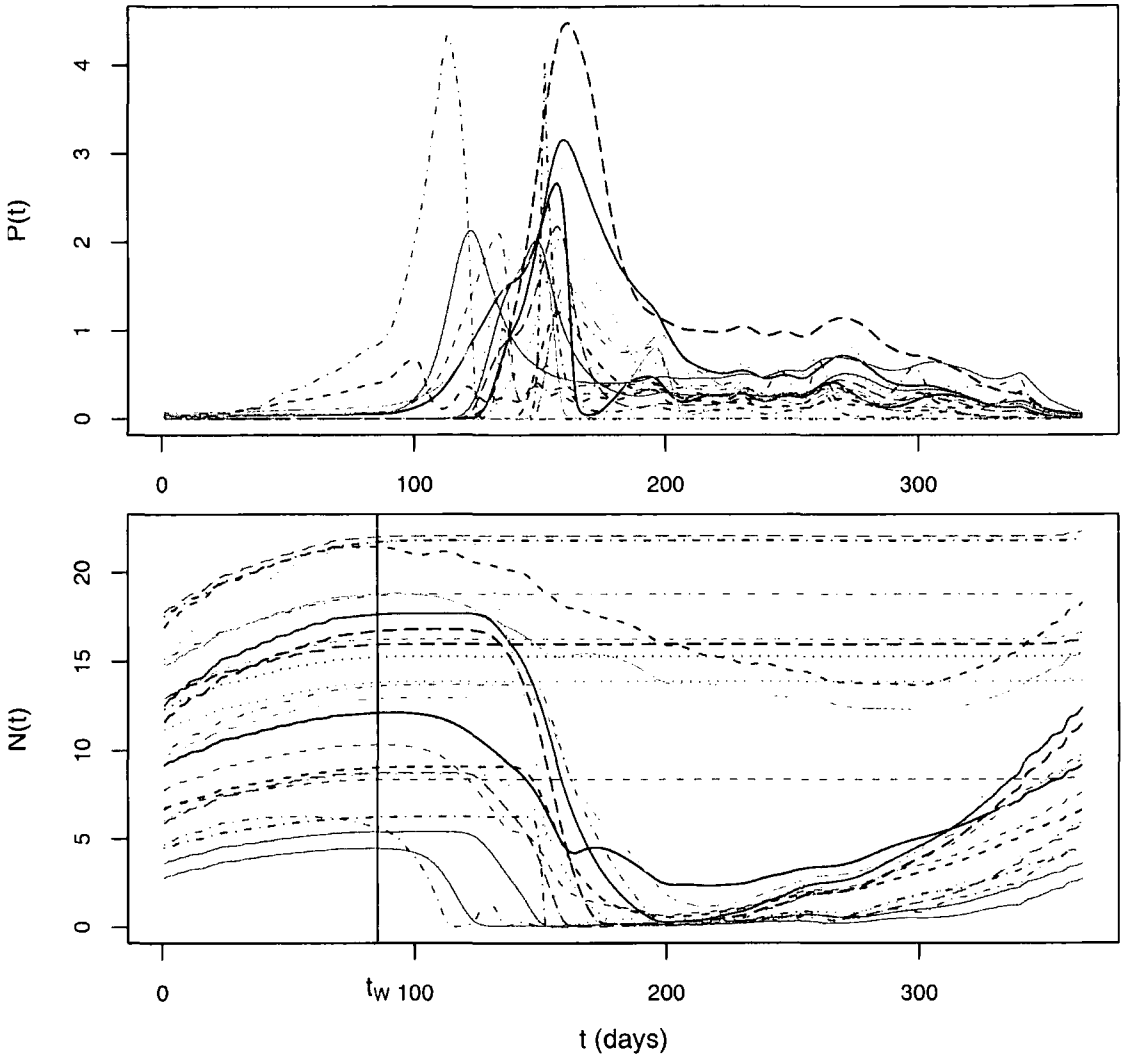


Figure 6.4: Station 1115: time series simulator output at different inputs for phytoplankton, $P(t)$, and nutrient, $N(t)$, plotted against time, t.

so that lower and upper bounds for x_i are mapped from $\tilde{x}_i \in [L_i, U_i]$ to $x_i \in [-1, 1]$:

$$x_i = \frac{1}{U_i - L_i} \left(2\tilde{x}_i - (U_i + L_i)\right). \tag{6.2}$$

For the most part, we work with the inputs in their transformed units, x_i, with the occasional exception, in particular in Chapter 7, when we use the symbols and original units, given Table 2.1, in linking our findings back to statements about the physical problem. Rescaling inputs necessarily requires to us to rescale derivatives generated by the simulator, $\partial s/\partial \tilde{x}_i$ and, wherever they are shown, derivatives are

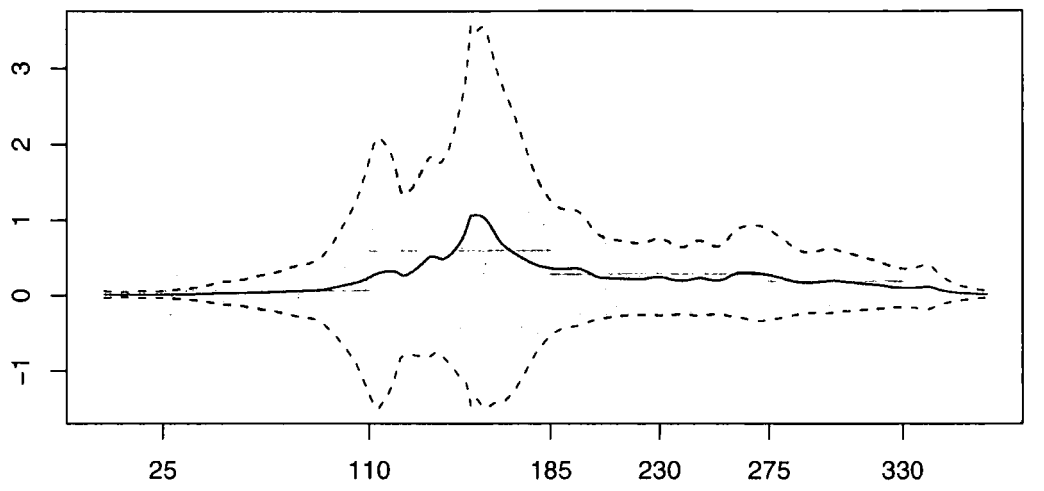


Figure 6.5: Station 1115: mean ± 2 s.d. phytoplankton simulator data. Black lines denote values for daily time series data and grey lines and rectangles denote values for each output of the interval re-parameterisation.

rescaled to correspond to inputs in their transformed units:

$$\frac{\partial s}{\partial x_i} = \frac{U_i - L_i}{2} \frac{\partial s}{\partial \tilde{x}_i}. \tag{6.3}$$

We began by choosing a fifth location, station 1115, as an exploratory site with a view to a preliminary analysis to aid our forming of prior beliefs at other stations. Station 1115 was chosen as it lay in the middle of the four calibration stations and we believed it would give us information about each of them. We re-parameterised output at station 1115 according to the intervals at station 1113; the first station at which we intended to fit a prior based on the exploratory analysis. Station 1113 was chosen because station 1115 lay slightly further away than it did the other three calibration stations so that any similarity between it and the remaining three sites would, we thought it reasonable to believe, be at least as strong.

We ran a 25-point Latin hypercube over the thirteen inputs in Table 2.1, taking a uniform prior distribution on $[-1, 1]$ for each input. Latin hypercube sampling (see Owen, 1992) returns a sample of n points from a space, χ, by dividing the prior distribution for $x \in \chi$ into n intervals of equal probability along each dimension, x_i, and sampling the grid boxes in such a way that each interval in each dimension

is sampled only once. In our case, the prior distribution is uniform over $[-1, 1]$ for each input dimension so that these intervals are distributed over an evenly spaced grid. The advantages of such a design is that it gives good coverage of the input space and lower dimensional projections of the sample retain this coverage.

Figure 6.4 shows the original untransformed time series data at the twenty five input points for phytoplankton (top) and nutrient (bottom) at station 1115. We see a large variability in the phytoplankton output in the middle of the year (between approximately 100 days to 200 days) with a much smaller variability at the start and beginning of the year. Figure 6.5 shows the mean $\pm$ 2s.d. for the phytoplankton simulator output before being re-parameterised (black lines) and under the interval parameterisation (grey horizontal lines and rectangles). We see the variation is much less for the transformed data which should help us to build an accurate emulator based on fewer runs (since observing output will tell us about outputs further away in the input space than under the original parametrisation) whilst still hopefully retaining enough variation for us to calibrate against (in our case, variation in the simulator output is still large relative to that in the physical observations under this re-parameterisation so that we can calibrate). The bottom plot in Figure 6.4 shows that nutrient output for any given input vector appears to be fairly smooth around t_w and hence modelling a single nutrient time point should not cause too many problems.

6.3.1 Choice of active inputs

For each input, we computed the mean and standard deviation of the twenty five derivative values at every seventh day of the original untransformed time series simulator output. Figure 6.6 plots the largest and second largest mean derivative and standard deviation derivative at each of these time points. The numeric plot symbols denote the input to which the derivative statistic corresponds and grey circles for the mean plots denote a negative value. The plots are designed to aid in spotting any general patterns which may influence our interpretation and selection of active inputs for the re-parameterised output. We see from the top row that phytoplankton tends to be most sensitive to changes in x_1, with this input registering the largest

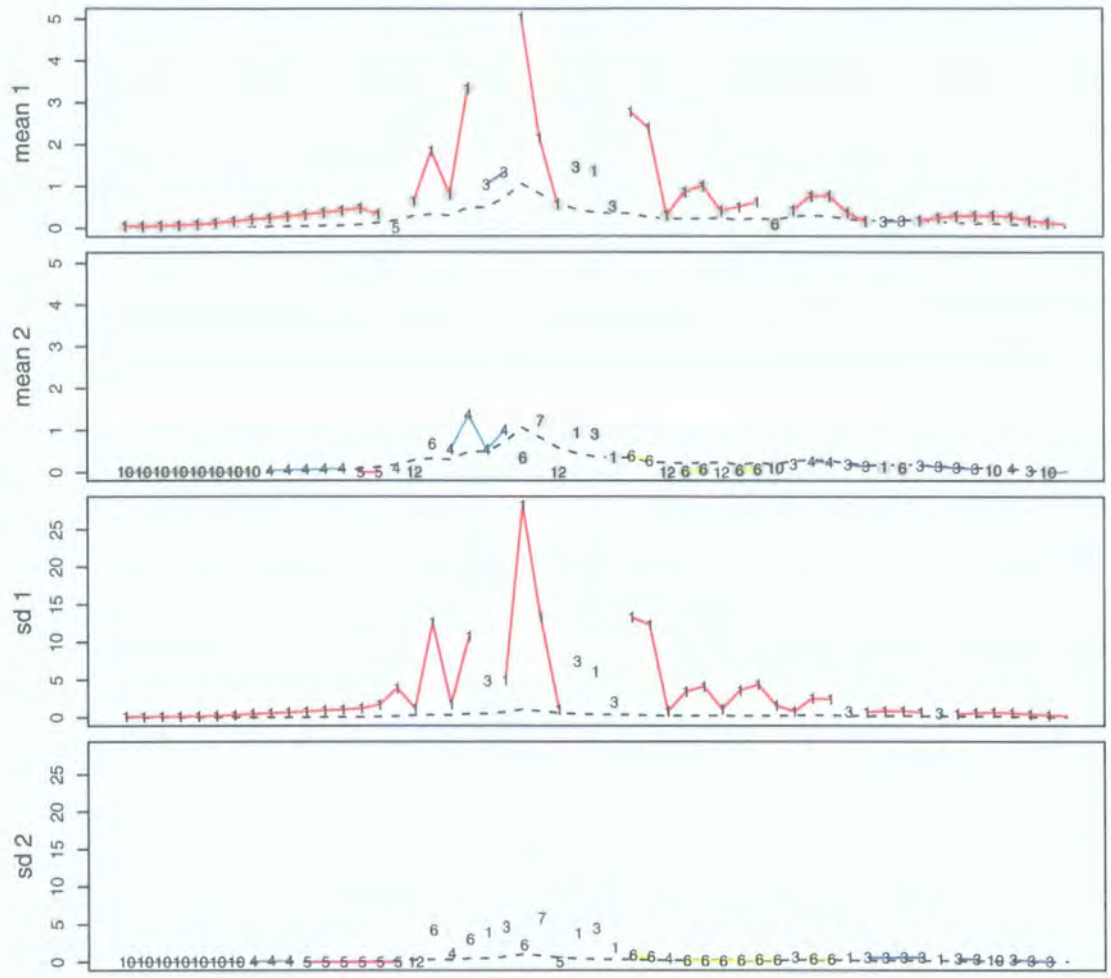


Figure 6.6: Station 1115: value of the largest and second largest phytoplankton input derivative mean and standard deviation, plotted at every seventh day of the original untransformed time series. The plot symbols are numbers corresponding to the input whose derivative is plotted and lines join values of inputs whose derivative appears at successive time points. For the mean plot, grey filled circles corresponds to derivatives whose values are negative.

absolute value in the mean and largest standard deviation at the majority of time points. In addition, x_3 registers as the largest absolute derivative in the mean and in the standard deviation for a small number of time points in middle of the year. Second order effects tend to be caused by x_4 and x_{10} early on and by x_3 later on, with x_6 also registering effects in the middle of the year.

Figure 6.7 shows the parameterisation of outputs for station 1115 together with

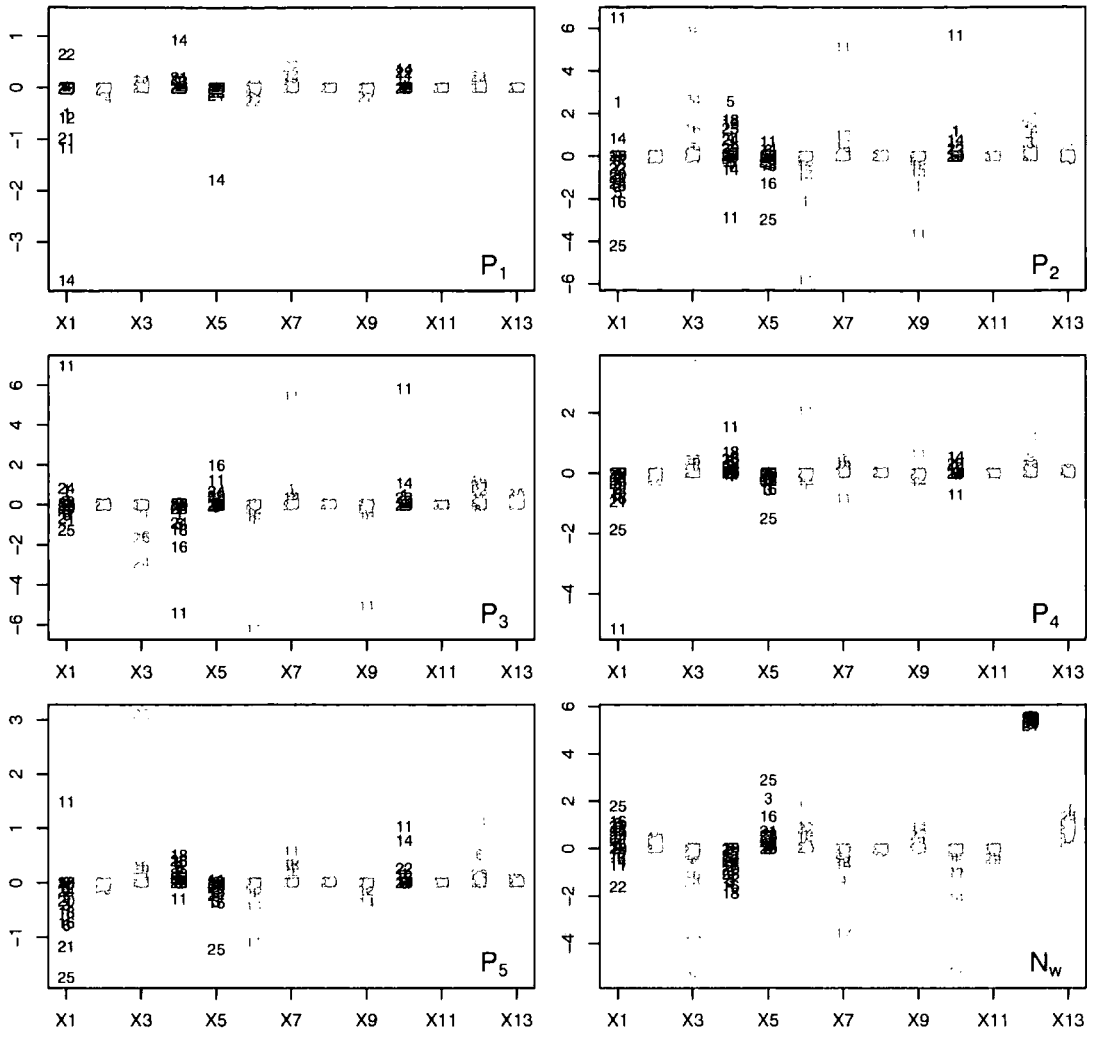


Figure 6.7: Station 1115: Derivatives of re-parameterised outputs with respect to each input computed at the twenty five input values. The set of active variables, $x^* = (x_1, x_4, x_6, x_{10})$ for $\bar{P}_{1115}$, and $x^* = (x_1, x_4, x_6, x_{13})$ for N_{1115}, are coloured black and grey vertical lines denote $\pm$2s.d. for each sample of derivatives.

plots of first-order input derivatives for each output at the twenty five input points in our hypercube design, with vertical lines denoting $\pm$2s.d. for each sample of derivatives. The plots provide an informal means to form various aspects of our prior beliefs. In looking for active variables for a given output, much of the information can be summarised by considering the sample mean and variance of each of the corresponding input derivatives. An input which has a collection of derivatives with

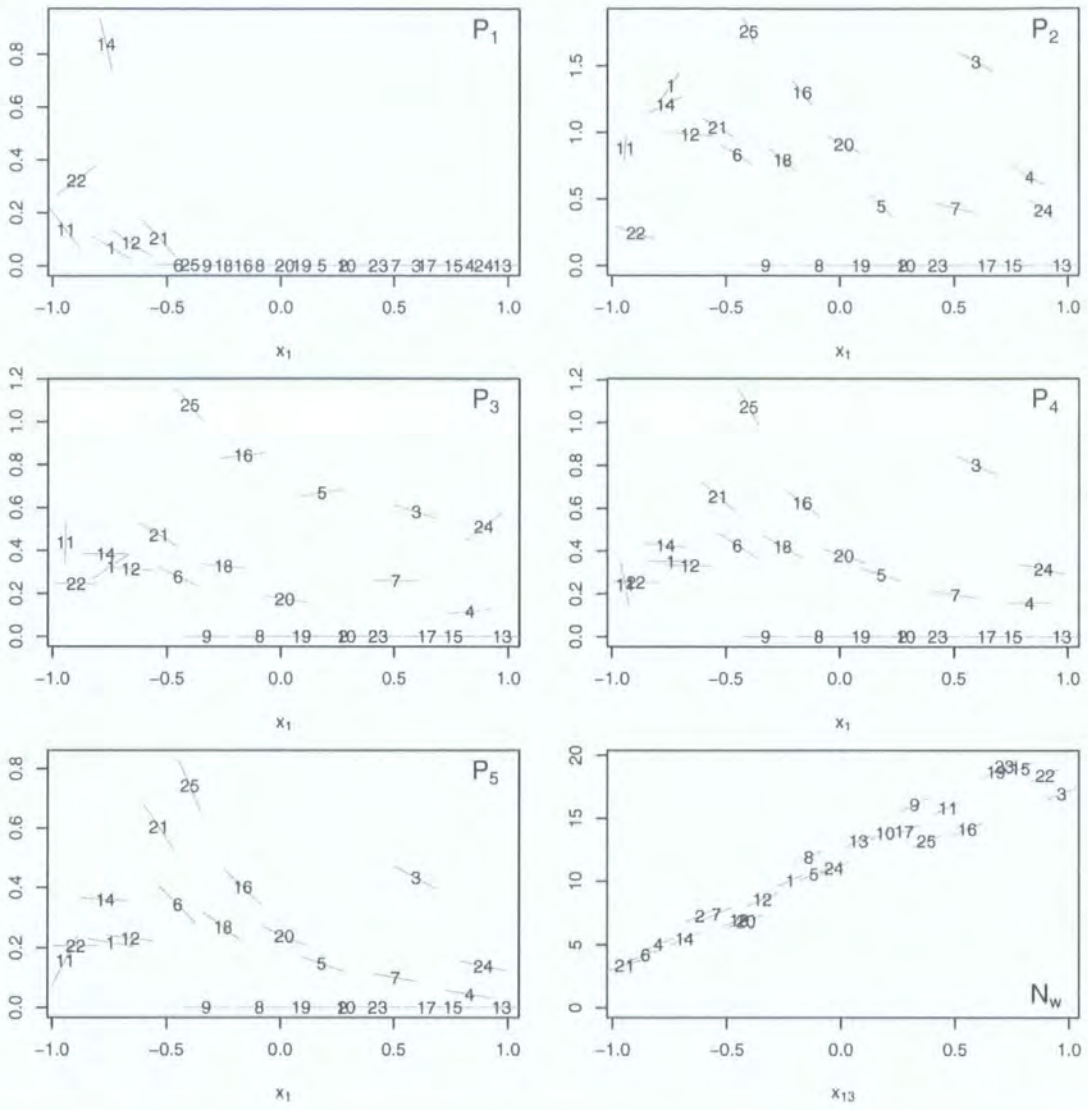


Figure 6.8: Station 1115: simulator output, $\bar{P}_i$ and partial derivatives, $\partial \bar{P}_i / \partial x_1$, $i = 1, \ldots, 5$, plotted against x_1 for the first five plots, and simulator output, N_w, and partial derivative, $\partial N_w / \partial x_{13}$, plotted against x_{13} for the final plot. For each output, function values are marked by their run number and derivatives by the gradient of the line passing through the corresponding point.

small variance and mean close to zero can be discarded as inactive and we might choose not to compute its derivatives in further runs. An input whose derivatives have mean far from zero but small variance can be considered as active, but we may wish also to drop derivatives calculation in further runs as derivatives appear constant throughout the space. Conversely, we may wish to calculate higher order derivatives in future runs of inputs whose derivatives exhibit high variance.

Figure 6.8 takes, for each output, the twenty-five function values plots them against what appears, from Figure 6.7, to be their most active input - x_1 for $\bar{P}$ and x_{13} for N_w - together with the twenty-five derivative values with respect to this input. A noticeable feature of the $\partial P_i/\partial x_1$ values is that they seem, in general, to be much more variable for smaller x_1 than for larger x_1. For N_w, we see the strong linear effect of x_{13} in the function values and in the constant $\partial N_w/\partial x_{13}$ values.

Output	Input 1	Input 2	Input 3	Totals
$\bar{P}_1$	x_1 (16,8,1)	x_4 (6,9,7)	x_3 (3,2,4)	(25,19,12)
$\bar{P}_2$	x_4 (8,1,7)	x_1 (7,15,2	x_3 (5,1,1)	(20,17,10)
$\bar{P}_3$	x_4 (9,0,4)	x_1 (3,12,3)	x_3 (5,1,2)	(17,13,9)
$\bar{P}_4$	x_4 (8,1,7)	x_1 (5,14,1)	x_3 (5,1,1)	(18,16,9)
$\bar{P}_5$	x_1 (8,11,2)	x_4 (7,4,5)	x_3 (5,1,1)	(20,16,8)
N_w	x_{13} (25,0,0)	x_{15} (0,13,5)	x_4 (0,3,9)	(25,16,14)

Table 6.1: Top three ranking inputs based on the number of runs at which an input scores one of the top three derivatives magnitudes. The numbers in brackets denote the number of runs at which each input scores the first, second and third highest derivative magnitude.

A noticeable feature in Figure 6.7 is a small number of large derivatives at each input. Thus the plots are good for spotting inputs with large derivatives, but these derivatives tend to warp the scale of the plot and make it hard to make a distinction between inputs at runs for which derivatives are smaller. A way round this is to produce versions of the plots with these extreme values removed. An alternative is to compare derivatives on a run-by-run basis: Table 6.1 does this by counting the number of runs at which each input has one of the top three derivative magnitudes. The numbers in brackets denote the number of runs at which each input scores the first, second and third highest derivative magnitude. For example we see that $\partial N_w/\partial x_{13}$ is the largest derivative for all 25 runs, backing up our previous findings. A word of caution about these numbers, however, is this: some runs are more important than others. In particular, an input which ranks as the largest derivative for the majority of input points, but for which all derivatives are small at these input points will score well here but we may not feel it to be active: x_{15} - which is the

second largest derivative of N_w at 13 of the 25 runs, may be one such input, based on Figure 6.7.

The derivative plots in Figure 6.7 are also rich in information about the kind of variation explained by each input. In our case, the plot of N_{1115} (bottom right) shows that the derivative with respect to x_{13} (N_{ref} in the original units) is consistently relatively large with little variation. This led us to suspect a strong linear effect from x_{13} dominating the variation for this output. Since there was no such consistent linear effect for the phytoplankton outputs, we decided to group our prior beliefs for the outputs into phytoplankton outputs and nutrient. For the nutrient output, N_{1115}, we chose the set of active inputs $x^* = (x_1, x_4, x_6, x_{13})$, marked black in Figure 6.7. Whilst there were some differences between phytoplankton outputs, given the small number of runs performed and the uncertainty inherent in linking it to the other sites, we decided not to break down our prior description further. We chose the active inputs $x^* = (x_1, x_4, x_6, x_{10})$ for $\bar{P}_{1115}$ since, between them, they were responsible for the majority of the variation in derivatives at the exploratory site outputs.

6.3.2 Assessing global prior mean and covariance parameters

In selecting and estimating terms in our model at the exploratory site, we adopted a model-fitting approach, fitting linear, quadratic and first-order interaction terms in the active inputs for each output using Ordinary Least Squares (OLS). Table 6.2 shows the R^2 values for linear fits with and without quadratic and interaction terms. From Table 6.2, we see that R^2 values for outputs in $\bar{P}$ are all much lower than for N_w and are all roughly the same order, although they tend to increase slightly for outputs corresponding to intervals further away from the centre of the year. All the phytoplankton R^2 values are much lower than for N_w.

In general, quadratic terms appeared to pick up less variation than interaction terms. Although the amount described by individual terms varied across different outputs, we chose to keep 'all linear terms' or 'all interactions' rather than hand-pick terms, in order to keep a reasonably flexible prior model which could be naturally

Output	$P_{1115,1}$	$P_{1115,2}$	$P_{1115,3}$	$P_{1115,4}$	$P_{1115,5}$	$N_{1115,w}$
R^2_{lin}	0.43	0.53	0.33	0.48	0.54	0.97
$R^2_{\text{lin+quad}}$	0.55	0.59	0.47	0.59	0.62	0.98
$R^2_{\text{lin+int}}$	0.83	0.66	0.54	0.69	0.78	0.98
$R^2_{\text{lin+int+quad}}$	0.93	0.72	0.66	0.80	0.87	0.99

Table 6.2: R^2 values from OLS fits at station 1115.

extended to other sites. Hence, we decided to keep linear terms for N_w and linear and first-order interaction terms for $\bar{P}$. This gave models of the form

$$\bar{P}_{1115,k} = \beta_0 + \sum_i \beta_i x_i + \sum_{i,j\ i\neq j} \beta_{i,j} x_i x_j + \sigma^2_{\epsilon,k}, \quad i,j = 1,4,6,10, \tag{6.4}$$

$$N_{1115,w} = \beta_0 + \sum_i \beta_i x_i + \sigma^2_{\epsilon}, \quad i = 1,4,6,13. \tag{6.5}$$

In order to improve the fits for the phytoplankton outputs, we experimented with including inputs which we had not chosen to be active based on our derivative plots, but this also had little effect and we were wary about spurious improvements because of the small number of runs used in building the OLS models. We also tried a log transformation of the simulator output data, and repeated the OLS model-fitting, but this did not result in any significant improvement in the R^2 values. Thus we kept the original models, believing them to represent a good description of global 'trends' in the phytoplankton outputs, but from which there was significant variation which would be described by higher order effects in the ϵ-surface (including the variation described by the dropped quadratic terms). Table 6.3 gives the OLS estimates for the coefficients in these models.

6.3.3 Assessing variance parameters for ϵ and δ

Craig et al. (2001) describe a 'spectral decomposition' approach to estimating σ^2_ϵ and σ^2_δ, in a Bayes Linear context, which does not require full distributional assumptions. The method estimates σ^2_ϵ and σ^2_δ by regressing the residuals of the corresponding OLS fit onto the eigenvalues of the model covariance of $\epsilon + \delta$ for a given choice of Θ as follows: Let e be the residuals from an OLS fit of $S = s(X)$ on X. Then $e =$

	β_0	β_1	β_4	β_6	β_{10}	$\beta_{1,4}$	$\beta_{1,6}$	$\beta_{1,10}$	$\beta_{4,6}$	$\beta_{4,10}$	$\beta_{6,10}$
P_1	0.07	-0.11	0.13	-0.01	-0.10	-0.21	0.07	0.21	-0.01	-0.15	0.06
P_2	0.63	-0.41	0.52	-0.03	0.18	0.02	-0.08	0.10	-0.10	0.33	-0.51
P_3	0.29	-0.23	0.17	-0.07	0.10	0.28	0.25	-0.05	-0.06	0.18	-0.30
P_4	0.28	-0.24	0.20	-0.12	0.15	0.23	0.24	-0.08	-0.13	0.22	-0.28
P_5	0.19	-0.21	0.13	-0.10	0.10	0.12	0.21	-0.08	-0.11	0.16	-0.19
	β_0	β_1	β_4	β_6	β_{13}	-	-	-	-	-	-
N_w	11.57	0.43	-1.31	-0.02	7.95	-	-	-	-	-	-

Table 6.3: Estimates for B coefficients from OLS fit.

$P(\epsilon+\delta)$ where $P = X(X^TX)^{-1}X^T$ is the projection operator from the OLS fit. Then $\text{Var}[e] = P\text{Var}(\epsilon+\delta)P^T = \sigma_\epsilon^2 PRP^T + \sigma_\delta^2 PIP^T$, where $R = \exp\{-(x-x')^T\Theta(x-x')\}$ is taken to be known. The two components of $\text{Var}[e]$ commute (to see this note that $P = P^T$ and, since P is a projection matrix, $P^n = P$ for $n \geq 1$) so that they can be diagonalised simultaneously in an orthonormal basis $u_1, \ldots, u_n$ with corresponding eigenvalues λ_i^ϵ and λ_i^δ. Then, for $v_i = u_i^T e$, we have that the u_i are independent with $\text{E}[v_i] = u_i^T\text{E}[e] = 0$ and $\text{Var}[v_i] = u_i^T\text{Var}[e]u_i = \sigma_\epsilon^2\lambda_i^\epsilon + \sigma_\delta^2\lambda_i^\delta$. Hence we can estimate σ_ϵ^2 and σ_δ^2 by regressing v_i^2 on λ_i^ϵ and λ_i^δ (with no intercept term).

Note that the method assumes that the value of Θ is known. In Chapter 4 we discussed the fact that estimating Θ values was difficult under a full Bayes analysis and, in fact, the same turns out to true in a Bayes Linear analysis. Craig et al. (2001) take a common θ for each input dimension and fix this value *a priori*, choosing this common value based on simulating random functions with a Gaussian covariance structure for different θ values, and choosing a θ which appeared to give realisations of roughly the desired variation. However, in Chapter 4, we illustrated with an example the potential that derivatives offer in estimating components of Θ. Here we offer a way to exploit this potential by extension of the method of Craig et al. (2001), which not only keeps true to the Bayes Linear spirit of dropping full-distributional assumptions, but is computationally cheap.

By considering the models for the derivatives of each output taken with respect to

each of its active variables

$$\partial s/\partial x_i = B\partial g/\partial x_i + \partial \epsilon/\partial x_i \tag{6.6}$$

together with our model variance for ϵ-derivatives which, from (4.13), is given by

$$\mathrm{Var}[\partial \epsilon(x)/\partial x_i] = 2\theta_i \sigma_\epsilon^2, \tag{6.7}$$

we proceed as follows:

1. Start with a prior guess for Θ (e.g. obtained from simulation).
2. Use the spectral decomposition method of Craig et al. (2001) to obtain estimates of σ_ϵ^2 and σ_δ^2 for each output.
3. Fit the global terms in (6.6), using e.g. OLS, to the corresponding derivatives generated by the simulator.
4. For each output, take the Residual Sum of Squares (RSS) of the fit in step 3 as an estimate of the left hand side of (6.7), and divide through by estimates of $2\sigma_\epsilon^2$ from step 2 to obtain estimates of the θ_i.
5. Iterate steps 2 - 4.

We applied this 'modified spectral decomposition method' in order to assess σ_ϵ^2, Θ and σ_δ^2 for each output of the exploratory site. For everything that follows, we take Θ to be a diagonal matrix and adopt the shorthand $\Theta = \mathrm{diag}(\Theta)$ for the vector of its diagonal elements. In choosing starting values for the elements of Θ, we took all components to be equal and chose a common value of $\theta = 2.5$. The value was obtained by simulating random functions on the range $[-1, 1]$, taken from a multivariate normal distribution with Gaussian covariance structure, for a range of different θ values, and choosing a θ which gave realisations of roughly the desired variation. In our case, $\theta = 2.5$ gave smooth realisations but also containing substantial quadratic and higher order effects.

Table 6.4 gives the results of steps 2 - 4 of the modified method at the exploratory site. Note that the method gives a estimate for each component of Θ for

Output	σ_ϵ^2	σ_δ^2	θ_1	θ_2	θ_3	θ_4
P_1	0.045	0.013	27.33	5.91	2.95	0.79
P_2	0.279	0.015	7.78	3.76	0.79	1.03
P_3	0.231	0.058	7.18	5.36	0.36	1.07
P_4	0.227	0.012	9.91	5.82	3.03	3.34
P_5	0.009	0.215	6.80	2.06	0.96	0.14
N_w	0.885	1.493	1.95	2.10	1.60	0.28

Table 6.4: Estimates from the modified spectral method at the exploratory site.

each output. Computational savings arise from outputs with common active inputs having common θ values for these inputs. For phytoplankton outputs, we decided to combine estimates over outputs to obtain a single Θ vector, but to allow different values for different components of Θ. In combining the estimates, we decided to ignore estimates corresponding to P_1 and P_5 since both these outputs had small σ_ϵ^2 estimates to which our method of Θ estimation, which involves dividing by σ_ϵ^2, is potentially very sensitive to any errors. The remaining Θ estimates for phytoplankton outputs were all the same order for any given component and we chose to combine them conservatively by setting each component to be the maximum value of that component over these outputs. This gave the estimate $\Theta = (9.91, 5.82, 3.03, 3.34)$ for $\bar{P}$ and $\Theta = (1.95, 2.10, 1.60, 0.28)$ for N_w. Finally, we plugged these estimates back into the spectral decomposition method to re-estimate σ_ϵ and σ_δ for the new Θ. Of the resulting estimates, there was very little change, with six of the twelve estimates registering no change at all.

Note that in general we could continue to reiterate the method, although we chose not to here because of the small changes and, in particular, because the estimates for σ_ϵ corresponding to P_2, P_3, P_4 were three of the six estimates which remained unchanged. A note of caution concerning an iterative method such as that outlined is this: unless the emulator residual is orthogonal to the regression terms (for example, with respect to a uniform distribution on x in our case), then the two parts of the model may begin to compete for variation because of the non-identifiability of the model. In particular, the residual may try to 'steal' variation from the regression

terms leading to a σ_ϵ term which is too large and a collection of Θ parameters which are too small. Since we made just one pass through the method here, this is not an issue; in general, however, it may be - and in fact it may be advisable to limit oneself to one iteration.

As mentioned previously, estimating Θ is a problem that has caused a great deal of difficulty in the literature. Access to derivatives, however, has offered us a way to estimate Θ which is simple owing to the form of the ϵ-surface and enables us to make prior distinctions between dimensions because the information contained within the derivatives is linearly related to Θ through (6.7). Diagnostics of emulator forecasting performance in Section 6.6 show nothing to contradict our belief that these estimates are good.

6.3.4 Design of simulator runs

In selecting the set of inputs at which to run the simulator, we set down a total budget of 200 runs across the four calibration stations and took a 200 point Latin hypercube over all thirteen inputs and then assigned each calibration station twenty five of these runs at random. A single hypercube was preferable to generating a 50 point Latin hypercube at each station as we wanted to maximise our coverage of the input space in our search for inputs which produce good matches at all four stations. Random assignment of these runs to each station was a simple and quick way to divide up the runs although better assignments are probably possible. Whilst Latin hypercubes offer space-filling designs which project well into lower dimensions, they do not guarantee orthogonality and can potentially result in highly correlated designs. Schemes to generate orthogonal designs, by contrast, are typically not space filling and this effect is exacerbated after projecting into lower dimensions. We were able to exploit the desirable properties of a Latin hypercube design whilst exercising control over the degree of orthogonality in a simple way as follows: we generated several candidate designs and computed the correlation matrix for each. For each input, we computed its maximum correlation with the remaining inputs, giving a total of thirteen correlations, denoted $C_1, \ldots, C_{13}$ after ordering from smallest to

largest. We then chose the design which minimised

$$wC_5 + (1 - w)C_{13} \text{ with } w \in [0, 1] \tag{6.8}$$

and investigated the effects of varying w, whose value reflects the relative importance of minimising the active input correlations relative to the remaining correlations. Finally, we permuted the columns of our final choice of hypercube so that the dimensions with the smallest five correlations corresponded to active inputs.

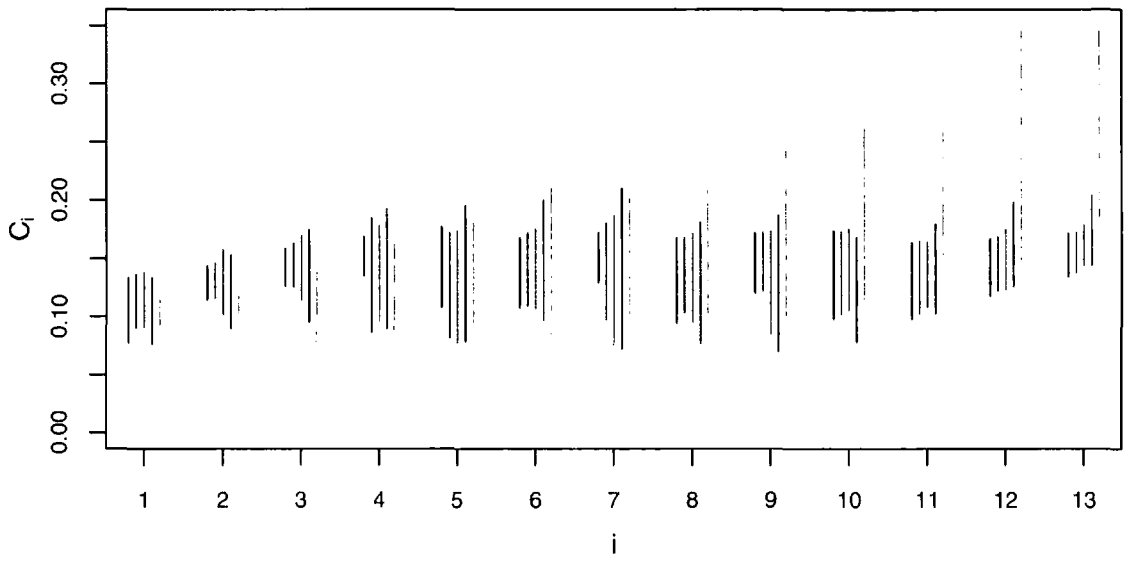


Figure 6.9: Mean ±2 standard deviation of $C_1, \ldots, C_{13}$ values based on 25 minimisations of $wC_5 + (1 - w)C_{13}$, with each minimisation over 10,000 randomly generated 13-dimensional 200-point hypercubes. Results for different values of $w \in (0, 0.25, 0.5, 0.75, 1)$ are shown from left to right for each C_i.

Figure 6.9 plots the means ±2s.d. for twenty five minimisations, each over 10,000 candidate hypercubes, for different values of w. We see firstly that, for w close to 1 (where we constrain C_5 only), that C_{13} shows a lot of variation in its value and can be relatively large. On the other hand, for smaller w, the constraint on C_{13} has the effect of constraining all C_5 reasonably well, but at the cost of not constraining the smaller C_i quite so much. Based on this, we generated a further 10,000 candidate hypercube and chose the design which minimised (6.8) with $w = 0.9$ chosen to direct most of the effort into minimising correlations between active inputs, but

precluding large values of correlations between inactive inputs. This gave a largest correlation involving an active input of magnitude $C_5 = 0.097$ in our final choice of hypercube. There are many sensible alternative criteria to (6.8) which could be employed to exert some control over correlations and this may be an interesting area for future work. We do not pursue this here, but note that employing some form of control over the degree of orthogonality of hypercube designs is probably desirable and can be achieved very cheaply (a sample of 10,000 hypercubes took approximately two minutes to generate and minimise on our computer). For some alternative approaches to improving Latin hypercube designs, see Koehler and Owen (1996) and references contained therein.

6.4 From exploratory analysis to prior emulator

The exploratory site had two main functions in assisting our emulation at station 1113. Firstly, it revealed active inputs and provided estimates of coefficients of the resulting model in these active inputs to aid with prior model building at station 1113. Secondly, again through the selecting of active inputs, it influenced our design of input points at which to run the simulator at station 1113, in order to refine our prior. A natural way to proceed in forming a prior at 1113, then, is through a combination of the estimates of the exploratory model-fitting in conjunction with informal assessments of our beliefs about the likely magnitude of difference between the two sites. We took the same form for outputs as at the exploratory site, given in (6.4), and took OLS coefficient estimates as the corresponding prior mean for the terms at station 1113. In forming variances for these coefficients, we considered for each coefficient a relationship of the form:

$$\mathrm{SD}[B_s] = a + b \times \mathrm{SD}[B_{s'}] \tag{6.9}$$

where we chose a and b to reflect our beliefs about the differences between the stations: a specifies what our uncertainty about a coefficient at station 1113 would be if we knew the value of the corresponding coefficient at 1115, $b \geq 1$ scales up uncertainty because OLS fitting assumes an uncorrelated error distribution which, in practice, does not hold. In the absence of strong information about a and b at this

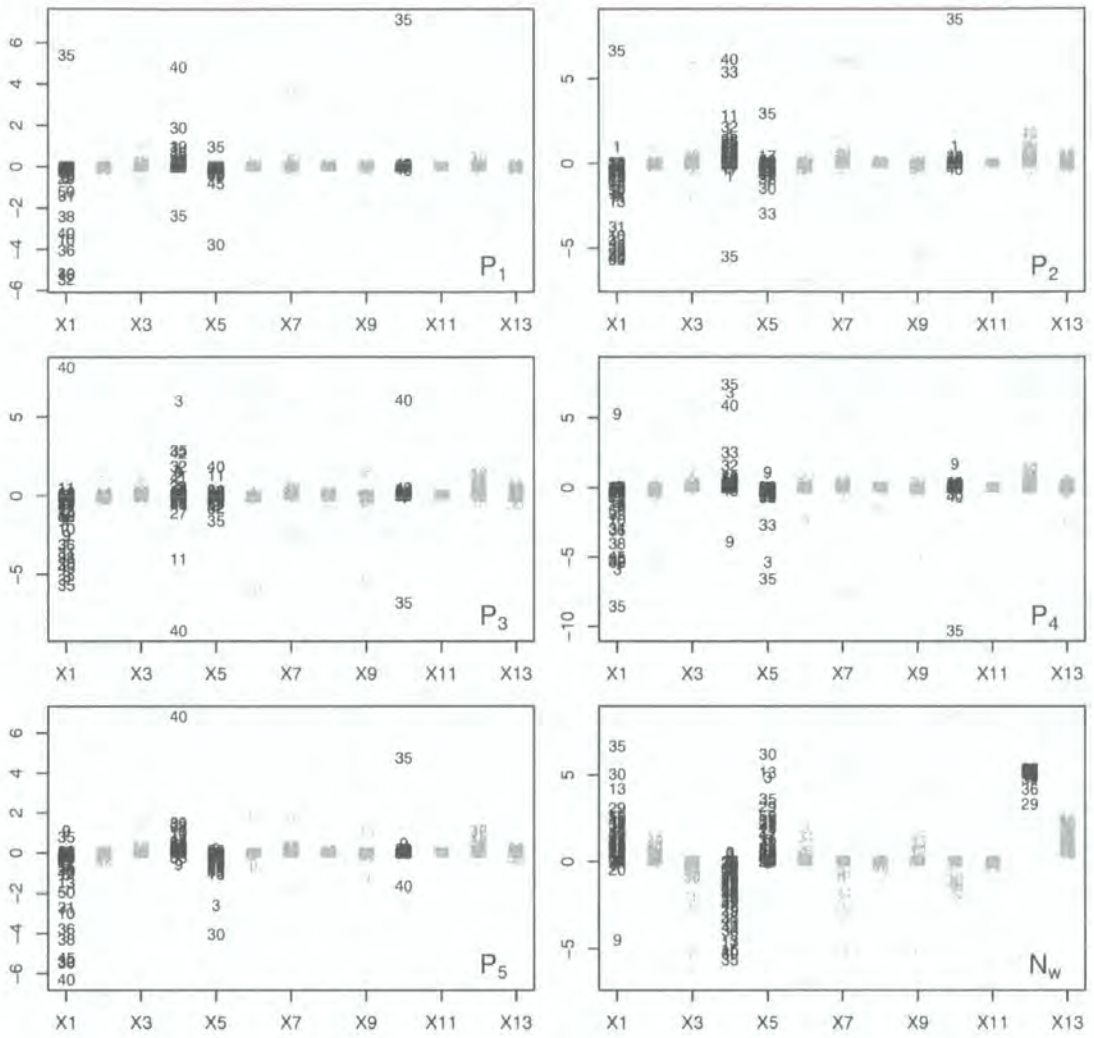


Figure 6.10: Station 1113: output parameterisation (top left) and derivatives of the re-parameterised outputs with respect to each input computed at the twenty five input values (remaining plots). See caption in Figure 6.7 for full description.

take, we chose to take them to be equal for all components of $\bar{P}$ and to choose them conservatively. An obvious way to learn about these values is to perform exploratory analyses at two sites. However, we chose instead to maximise our run budget for updating and scale uncertainty conservatively here (We can scale less conservatively when we link to and form priors for subsequent sites). In general, a more complicated linking relationship than (6.9) may be appropriate. For example, if the coefficients are on vastly different scales, we may want to scale with the quantity itself as well;

this was not the case here however and so (6.9) was felt to be sufficient.

Figure 6.10 gives the derivative plot for twenty five input points assigned to station 1113 at random from our 200-point hypercube, which we use as a diagnostic for checking aspects of our prior beliefs. In general, if for some station, the plots look very different to those at the exploratory site, we might for example perform OLS fits on the simulator output at the station and, in some cases, add extra terms into our prior emulator model, although we did not feel this to be necessary in this case. Similarly, our hypercube design is flexible in being easy to sample over extra active inputs dimensions for the seventy five remaining points if we do find extra or different active inputs. For N_{1113} we saw much the same behaviour as at the exploratory site with a large linear effect from x_{13} indicated. For the $\bar{P}_{1113}$, the plots reinforced our choice of active inputs as the four were active across most outputs and did not appear to contradict our prior groupings.

6.5 Resolving emulator uncertainty at a single site

We are interested in the additional reduction of emulator uncertainty caused by evaluating derivatives. For the calibration problem in hand, we are additionally interested in the effect this reduction has on our ability to calibrate the model, which we explore in Chapter 7. In making the comparison, we introduce the subscript notation X_n to denote the first n input vectors in X, with S_n and $\nabla_{x^*} S_n$ the corresponding simulator output and derivatives respectively. We evaluated our adjusted beliefs, for each output, on an evenly spaced 15^4 grid representation of the possible values of the four corresponding active inputs. This equates to $15^4 = 50,625$ evaluations for each output, which we can compute in less than one minute, demonstrating the strength of the emulator method. To compare variance reduction with and without derivatives, we considered the mean adjusted standard deviation,

$$\bar{\mathrm{SD}}_{D_n} = \operatorname*{mean}_x \mathrm{SD}_{D_n}[s(x)] \tag{6.10}$$

and the maximum adjusted standard deviation,

$$\hat{\mathrm{SD}}_{D_n} = \max_x \mathrm{SD}_{D_n}[s(x)], \tag{6.11}$$

both taken over the 15^4 grid representation of the active input space.

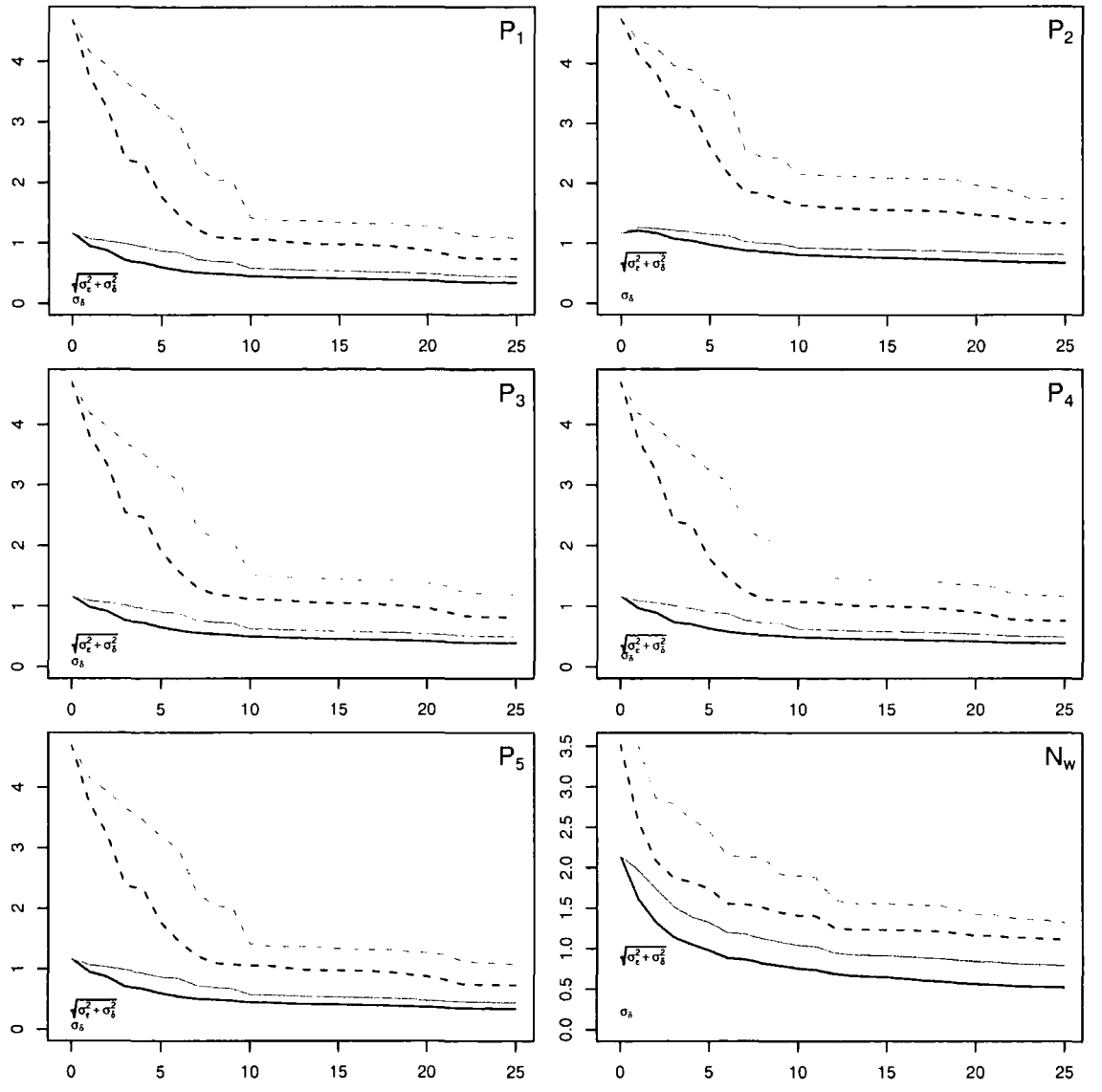


Figure 6.11: Adjusted standard deviation against number of simulator runs with derivatives (black) and without derivatives (grey) for outputs at Station 1113. Solid and dashed lines correspond respectively to the mean and maximum standard deviation taken over the input space. The light grey rectangle is defined by the horizontal lines $\sqrt{\sigma_\epsilon^2 + \sigma_\delta^2}$ and σ_δ, shown for reference.

Figure 6.11 shows $\bar{\mathrm{SD}}_{D_n}$ and $\hat{\mathrm{SD}}_{D_n}$ against the size, n, of the input set, X_n, for the outputs at station 1113, comparing the cases when derivatives are and are not included. The vectors in X appear in the order that they were selected at random from the 200-point hypercube so that X_{n+1} is equivalent to X_n together with an

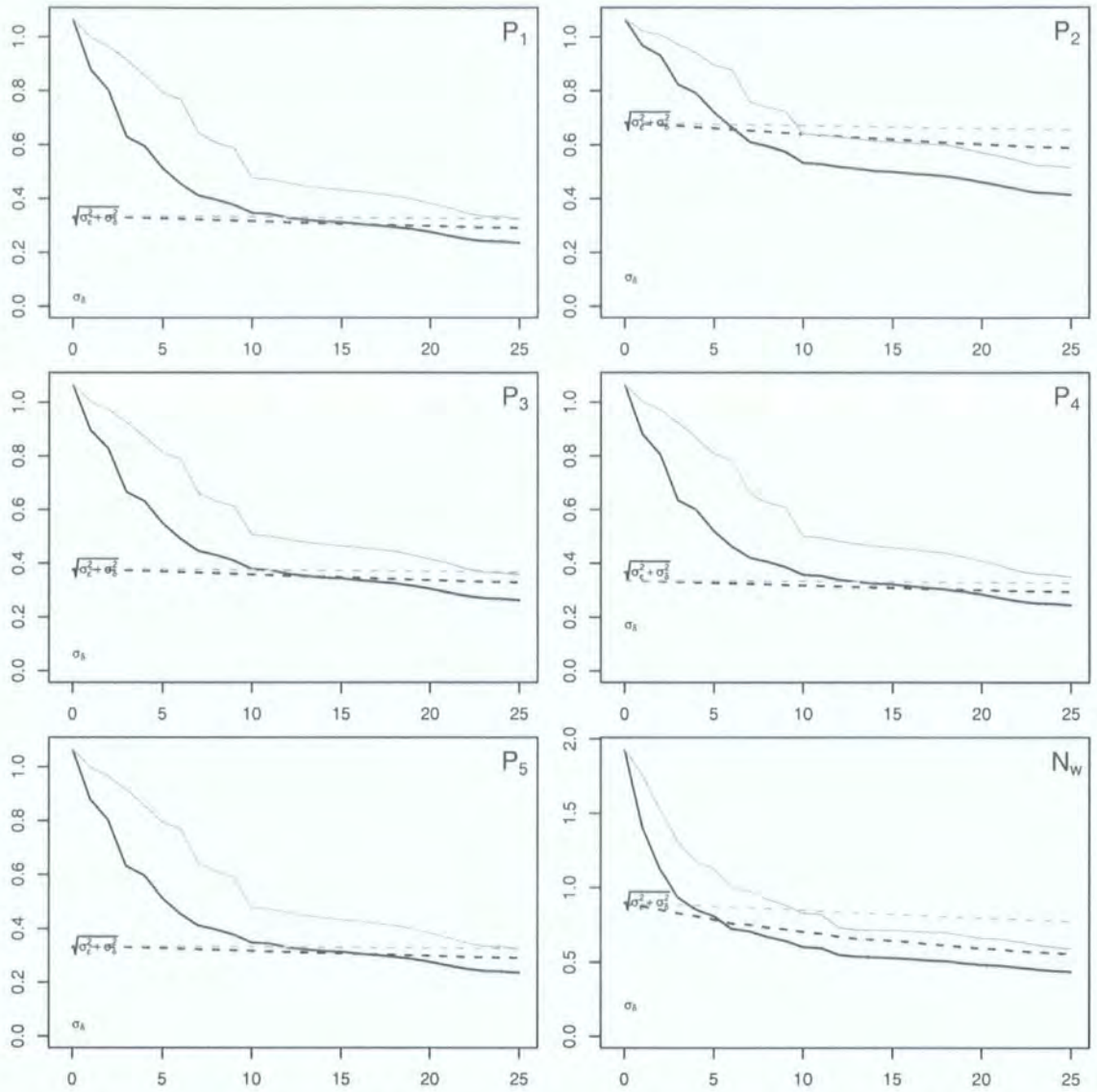


Figure 6.12: Station 1113: Separate components of adjusted standard deviation with derivatives (black) and without derivatives (grey). Solid lines correspond to $\mathrm{SD}_{D_n}[Bg(x)]$ and dashed lines to $\mathrm{SD}_{D_n}[\epsilon(x)]$. The light grey rectangle is defined by the horizontal lines $\sqrt{\sigma_\epsilon^2 + \sigma_\delta^2}$ and σ_δ, shown for reference.

input vector randomly chosen from the remaining input vectors and does not, for example, correspond to an ordering of the most informative runs.

For each output, the σ_δ line corresponds to the irreducible part of the variance and is thus a lower bound on the overall standard deviation, whilst the $\sqrt{\sigma_\epsilon^2 + \sigma_\delta^2}$ line denotes the level of standard deviation at which most of the global variance has

been resolved and after which we learn mainly about the ϵ-surface. In both plots we see that both the mean and maximum are significantly lower when derivative information is included (black) than when it is not (grey).

We see that, for N_w, the standard deviation decreases much faster than for the phytoplankton outputs for small n, which is a consequence of the smaller number of global terms in B leading to quicker resolving of global variation, which is the dominating variance for small n. This is confirmed also by the plot of separate part of the variance surface, given in Figure 6.12. In the bottom plot, the means are furthest apart after five runs and, after twenty five runs, the mean without derivatives has caught up quite a lot of the difference (although the maximum still has some way to go). In contrast, in the phytoplankton plots, the slower decrease in standard deviation means that, after twenty five runs, the difference between the two is just about levelling off at its maximum value so that we feel the benefit from including derivatives over the whole range.

We also investigated percentiles of the distribution of standard deviations such as the 50th and 95th percentiles. We found that the 50th percentile remained very close to, although slightly lower than, the mean and the 95th percentile remained approximately half way between the mean and maximum, giving an indication of the order of positive skew of the distribution.

The benefit of including derivatives can be thought of in terms of the horizontal (for a given number of runs with derivatives, how many runs do we expect to need to perform to obtain the same variance reduction without derivatives?) and the vertical (for a given number of runs, what is the additional reduction in variance achieved by observing derivatives?). Figure 6.11 shows us that, when we reduce largely global variation, the vertical gain can be much higher whereas, as we resolve variance largely in the ϵ-surface, the horizontal benefits are the greater. Considering vertical gain, for the outputs in Figure 6.11, the additional reduction in mean variance for N_w is at its greatest after five runs at 44.5%, after which it falls gradually to 31.2% after twenty five runs. For $\bar{P}$, the additional reduction in standard deviation climbs slowly as we increase the number of simulator runs and is not far off levelling out at 30.3% after twenty five runs, after which we expect it to begin falling slowly

again. As a measure of horizontal gain, we took the number of runs based on derivatives at which the mean variance surpassed that obtained by $n = 25$ runs without derivatives, which we found to be $n = 8$ and $n = 6$ for outputs $\bar{P}$ and N_w respectively. This corresponds to horizontal gains by factors of 3.1 and 4.2 for these two outputs, which represents a substantial saving in effort given our expected increase in cost by a factor of 1.8 in generating them.

6.6 Simulator Diagnostics

For each output, we considered one-step diagnostics for each of the corresponding simulator runs as follows: for the $(n+1)$th input choice, x_{n+1}, we computed $\mathrm{E}_{D_n}[s(x_{n+1})]$, the adjusted expectation based on observing output, $D_n = (S_n, \nabla_{x^*} S_n)$, at the first n input vectors and compared this to $s(x_{n+1})$, the value obtained from running the simulator at x_{n+1}, after dividing each by $\mathrm{SD}_{D_n}[s(x_{n+1})]$, the corresponding forecast standard deviation. Figure 6.13 shows diagnostics for outputs at station 1113 based on function values and Figure 6.14 plots the same quantities when derivatives are included in the updating. In both plots we see a general decrease in uncertainty as we learn about the simulator and observations generally falling within ± 3 forecast standard deviations of the forecast mean. The only notable exception is $\mathrm{E}_{D_{29}}[s(X_{30})]$ which underestimates the collection of unusually large phytoplankton values at X_{30}, and affects both function value and derivative plots.

To compare derivatives and function values more directly, we considered the quantity $\Delta_{S_n} - \Delta_{(S_n, \nabla_{x^*} S_n)}$ where Δ_{D_n} is defined by

$$\Delta_{D_n} = \frac{\mathrm{E}_{D_n}[s(x_{n+1})] - s(x_{n+1})}{\mathrm{SD}_{D_n}[s(x_{n+1})]} \tag{6.12}$$

Figure 6.15 plots these values together with a running average of the scores for each n. The expected value of this quantity is zero and so we expect, in particular, the running average to be close to 0 for large n; from the Figure 6.15 this seems to be the case.

In general we do not expect that the derivative-based emulator will do better on every individual forecast, but if for example the inclusion of derivatives leads to consistently worse forecasting, then this provides a diagnostic concerning the model

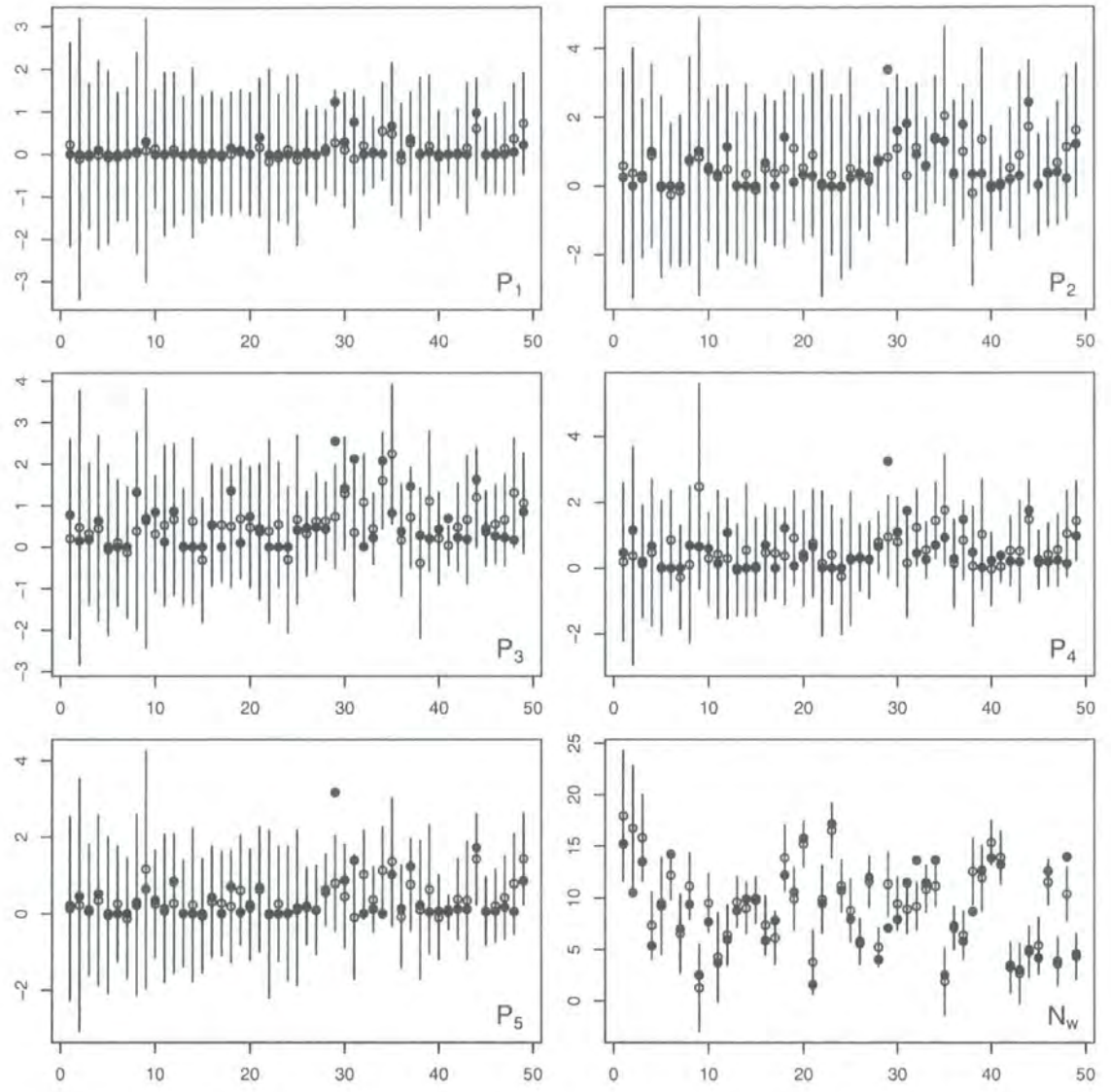


Figure 6.13: Forecast diagnostics based on function values for the outputs at station 1113 plotted against the number of runs, n. Filled circles correspond to $s(x_{n+1})$, the value observed when running the simulator at x_{n+1}. Unfilled circles correspond to $\mathrm{E}_{D_n}[s(x_{n+1})]$ and vertical lines show $\pm 3\mathrm{SD}_{D_n}[s(x_{n+1})]$ for $D_n = S_n$.

specification and may lead us to revise our beliefs. Hence the greatest benefit from including derivatives may not be that it leads to better forecasting; rather that, in the cases that it leads to a worse forecasting performance, a model misspecification is revealed about which we might otherwise have been blissfully unaware.

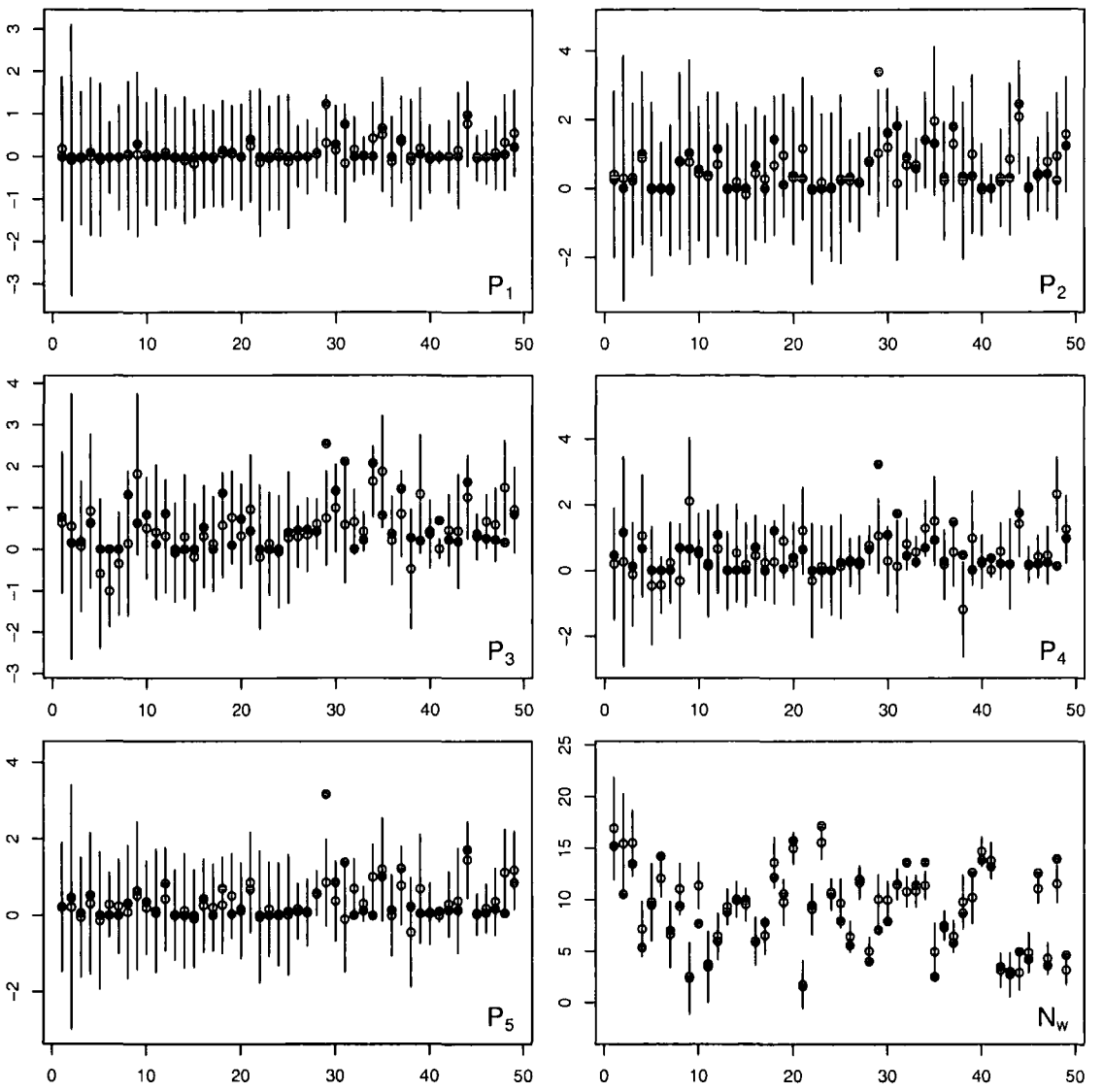


Figure 6.14: Forecast diagnostics based on function values and derivatives for the outputs at station 1113 plotted against the number of runs, n. See Figure 6.13 for details ($D_n = (S_n, \nabla_{x^*} S_n)$).

6.7 From single to multiple sites

The process of building emulators at multiple sites is much the same as at a single site: we transform the data at the exploratory site onto the intervals corresponding to the station under consideration and form a prior based on the same fitting approach of this transformed data. However, some additional features arise for consideration in choosing priors for remaining sites, in particular relating to our in-

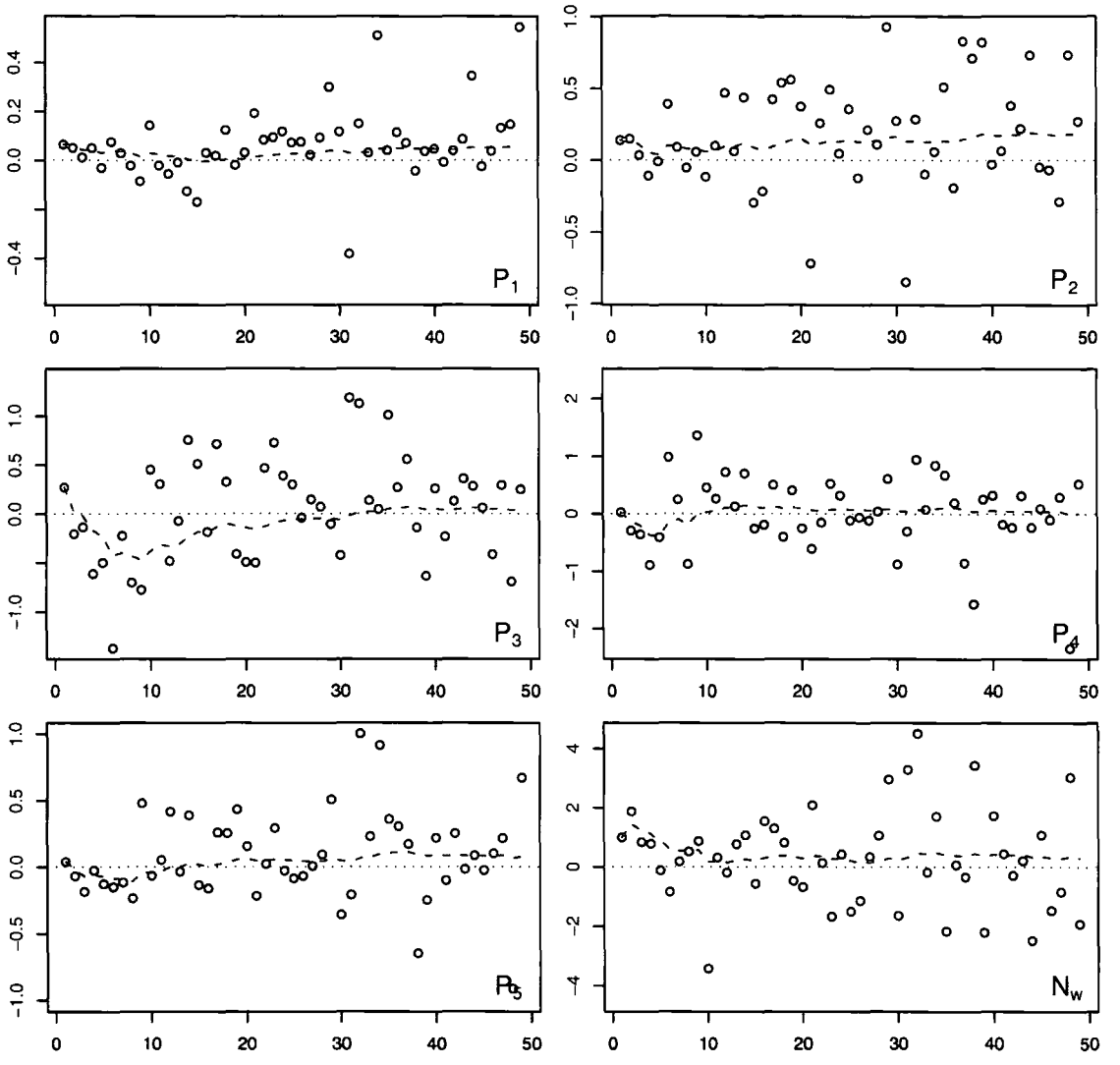


Figure 6.15: Plots of $\Delta_{S_n} - \Delta_{(S_n, \nabla_{x^*} S_n)}$ for station 1113 (See Eqn. 6.12). The red dotted line shows Derr=0 and the blue line gives the running average of the first n Derr scores.

formal adjustments of uncertainty. If we consider, for example, emulating a second site, station 1116 say, then as well as the prior emulator based on our exploratory analysis at station 1115, also available to us is the posterior emulator at station 1113. To form prior expectations at other sites, we can build OLS models based on the exploratory site in exactly the same way as at station 1113. Parameterising the exploratory data by the interval ranges belonging to the station we wish to emulate, we believe, captures the common variation over these intervals, described by the simulator time dynamics. The difference between prior and posterior beliefs

at station 1113 potentially offers valuable information when considering our prior variances at the second emulator station (that is, relating to the how we rescale prior uncertainty from the exploratory site to the second emulator site) in giving us an idea about the scale of warping of the common variation by differences in forcing functions.

In particular, taking the results at the exploratory site and using them in building our prior for the first emulator site essentially involved a scaling up of uncertainty according to our beliefs about the relationship between the two. Owing to a lack of knowledge about the magnitude of the difference between stations 1115 and 1113, the scaling we chose was one which, we believed, was conservative, leaning towards the upper end of what we thought the variance of the difference may be. In practice, it would seem to make sense to scale conservatively when linking early emulators and reduce this scaling (and presumably the number of runs performed) for prior emulators at later stations based on what we learn about the differences between stations. Informal updating of our beliefs about the difference between sites could be reviewed at each stage of a sequential emulation of sites. However, here we considered only a review from the first to second emulation site, considering that this link would encapsulate the bulk of the revision. We reviewed diagnostics for each site, given in Appendices A.1 and A.2, similar to those used in emulating station 1113, to make sure that there was no evidence obvious conflict between prior of run data at the sites.

We could potentially improve our understanding of the relationship between sites by including some replicates of the input points at which we ran the simulator for the earlier stations although we refrained from doing this in favour of maximising coverage of the input space within our budget. Also we could build a multivariate covariance structure over outputs and stations: this is a big area for future work related to the physical problem.

Chapter 7

Calibrating the model

In this chapter, we perform a calibration experiment on the simulator. We begin by introducing and justifying an 'implausibility' approach to calibration and then calibrate the model at several locations using the approach. We investigate the knock-on effect the inclusion of derivatives has on our ability to calibrate, by comparing calibration based on function values with that which also uses derivatives. Finally, we obtain a calibrated region for the simulator using the full collection of function values and derivatives as our best answer to the calibration problem.

We follow Craig et al. (1996, 1997) who develop an approach to calibration using the notion of "implausibility". An implausibility-based approach is a form of global search which seeks to identify classes of solutions by ruling out implausible inputs, where an implausible input is one which is "unlikely" to have given rise to the physical observations under some appropriate metric. In order to define such a metric, we need first to link simulator output, $s(x)$, to the physical observations, $\hat{y}$, which we do in the following section.

7.1 Linking simulator and physical system

In linking scalar simulator output, $s(x)$, to the physical observation, $\hat{y}$, we proceed in two stages. Firstly, we link the system, y, to physical observations, $\hat{y}$, by the relationship

$$\hat{y} = y + e \tag{7.1}$$

where e is the observation error, taken to be independent of y with expectation 0 and variance σ_e^2. Secondly, we link the simulator output, $s(x)$, with the system, y, via the following equation

$$y = s(x_0) + \eta \tag{7.2}$$

Eqn. (7.2) assumes the existence of a unique but unknown value x_0 for which we assume $s(x_0)$ summarises all the information about y contained within evaluations of the simulator; in other words, if we were to run the simulator at x_0, we would learn nothing more by running the simulator elsewhere. This judgement is expressed by taking the discrepancy, η, between this representation and the system to be independent of each $s(x)$ and of x_0, with expectation 0 and variance σ_η^2.

In terms of the plankton model, we assume that there is a single setting of the input parameters for which the associated output is sufficient for the plankton model in summarising everything we can learn from the plankton model about plankton cycles in the sea (at the scale we consider). The discrepancy η is a measure of how closely this best model output reproduces reality. The fact that we consider η to be independent of the simulator means that, for example, we don't allow for situations where one choice of input parameters produces good simulator output at some time points only and a different choice of parameter settings produces good output at other time points only. In practice, it is hard to justify this assumption - although it is currently the norm. An alternative would be to allow the discrepancy to depend on the value of x_0 (see e.g. Rougier, 2004) although again we would not be confident in specifying the exact form of such a dependency. Another approach would be to make these assumption not in linking the plankton model to reality, but to link the model to a 'better' plankton model. This 'better' plankton model would be closer in some sense to reality and the linkage would then perhaps be more justifiable. Goldstein and Rougier (2004) outline an approach in this vein, in which this consider a hierarchy of increasingly detailed simulators.

7.2 Implausibility in a single output dimension

We now proceed with a hypothesis-testing type argument, developing a measure of how likely, based on our beliefs, a given input point, x, is to produce a good match under the assumption that x is the best input, x_0. If the measure suggests this to be unlikely, then we consider this to be evidence that x is *implausible* as x_0.

Under the hypothesis that $x = x_0$, (7.1) and (7.2) together yield

$$\begin{aligned} \mathrm{E}_D[\hat{y}] &= \mathrm{E}_D[s(x)] \\ \mathrm{Var}_D[\hat{y}] &= \mathrm{Var}_D[s(x)] + \sigma_\eta^2 + \sigma_e^2 \end{aligned} \tag{7.3}$$

We then evaluate the implausibility, $I(x)$, of a point, x, as

$$I(x) = \frac{|\mathrm{E}_D[s(x)] - \hat{y}|}{\sqrt{\mathrm{Var}_D[s(x)] + \sigma_\eta^2 + \sigma_e^2}} \tag{7.4}$$

Large values of $I(x)$ indicate that there is a large standardised difference between the expected simulator output, $\mathrm{E}_D[s(x)]$, and the corresponding physical observation $\hat{y}$ and thus that x is unlikely to be a good choice of calibration value x_0. Small values can indicate one of two possibilities. First, $\mathrm{Var}_D[\hat{y}]$ may be large, so that we are too uncertain to rule the input x out. Second, the expected difference between simulator and physical observation may be small so that x lies within the class of acceptable matches. The presence of the separate parts of the denominator in (7.3) can be intuitively understood as follows: the larger $\mathrm{Var}_D[s(x)]$, the less certain we are about the value the simulator would produce if run at x and so the less willing we are to rule x out; the larger σ_e^2 the less useful the physical data is for calibration; the larger σ_η^2 the less able we believe the simulator to be of reproducing the physical system - and hence the larger the difference between simulator output and historical observations can be before we rule out the input. In our example, as a simple rule of thumb, we rule an input x as implausible if $I(x) > 3$; for comparison, the *three sigma rule* (Pukelsheim, 1994) states that for any unimodal density, at least 95% of the probability lies within three standard deviations of the mean. Where appropriate, we check that our results are not too sensitive to the choice of cut-off to make sure,

for example, that altering the cut-off value slightly does not vastly alter the number of points ruled implausible.

Finally, we note that the "ruling out" approach to calibration underpins the philosophy in our choice of parameterisation of phytoplankton output in section 6.2; that we do not necessarily believe that the minimum fit based on the interval means for phytoplankton will produce the best representation, rather that if an input does not produce an acceptable match to the means, then we wish to discard it from further consideration with the best representation lying *somewhere* within the remaining input region.

7.3 Combining Implausibilities

We combine implausibilities, $I_1, \ldots, I_k$, corresponding to scalar components of a vector of outputs, $Y = (Y_1, \ldots, Y_k)$, using

$$I_Y(x) = \max_i I_i(x) \tag{7.5}$$

reflecting our desire that an input be ruled implausible if it is implausible for any of the outputs (See Craig et al., 1997, for some alternative measures). In visualising plausible regions in the input space, we consider plots of lower dimensional projections of the implausibility surface given by

$$I_Y(x') = \min_{x''} I_Y(x) \tag{7.6}$$

where x', a subset of x, is the set of the components to be plotted and x'' is the set of remaining components over which we minimise. Such a projection enables us easily to rule out regions based on the plot, since for a projected input to be implausible it must be implausible for all combinations of the omitted components.

7.4 Bayesian vs Bayes Linear calibration

In a fully Bayesian approach, probability distributions must be specified for e and η in (7.2) and (7.1) and, in theory, a posterior distribution for x_0, conditional on the simulator run data and physical observations, derived. In practice, however, the

posterior is not available in closed form and computationally intensive numerical quadrature routines must be used which quickly become infeasible for simulators which are slow or contain large numbers of inputs (See Kennedy and O'Hagan, 2001a, who obtain the posterior mean for x_0 after approximating prior beliefs to be normally distributed and applying such techniques and give a discussion of the limitations of such an approach). In our case, beliefs about the simulator are characterised by a Bayes Linear belief structure which prohibits such a full Bayes analysis. It is worth recalling that one of our main motivations for a Bayes Linear approach is that the complexity of the simulator makes us uncomfortable about specifying full distributional assumptions, since we are unclear about how accurately such assumptions reflect our beliefs. In general, experts tend to be less clear about their beliefs relating to quantities linking the simulator to the historical observations - in particular, the η-surface - than they are about their beliefs concerning the simulator, adding further weight to the case for a Bayes Linear approach in general, since the ultimate aim of any emulation will be to use it to make inferences about the real world.

7.5 Choosing σ_e and σ_η

In order to compute implausibilities, we first had to elicit σ_e and σ_η for each output. The observational error estimates for σ_e relating to phytoplankton components were specified by the expert and derived from two sources. Firstly, an estimate of variation caused by mesocale eddy activity was obtained from sample variances of pixels within a 150km radius of the corresponding station. Secondly, the error in the SeaWiFS chlorophyll pixel estimates was taken to be such that 3 error standard deviations corresponded to (35%) of the observed value, based on the SeaWiFS target to achieve 95% of the data within ($\pm$35%) of in situ measurements of the same quantities. Analysis in O'Reilly et al. (2000) appears to show these errors are uncorrelated overall, but we expect there to be some spatial and temporal correlation between measurements, which isn't tested for in the analysis, and so we chose, conservatively, to take the interval mean of these observations to have 3.s.d also of ($\pm$35%). (For

an interesting exploration of more complicated measurement error structures, see Buck et al., 1996). No data were available to us of the variablity of the nutrient observations and we took these observations to have variance 1 at each station, based on the estimate of the expert.

For the discrepancy term, σ_η, we began by considering residual mean-squared values between simulator and physical observations: Hemmings et al. (2004) calculated residual mean squared values for each station by summing the squared difference between physical observations and his 'best' output, and found values to be between 2 and 4 observational standard deviations for chlorophyll and between 0 and 2 for nutrient. We expected discrepancy estimates of roughly the same order to be appropriate, although not necessarily identical since Hemming's calibration method differed in several ways from ours, using a different output parameterisation, a different set of stations over which to calibrate and constructing a 'misfit function' - essentially an implausibility measure (different to ours) - which he then attempts to minimise. With this in mind, we took individual components of σ_η to be equal to the corresponding components of σ_e, whilst investigating the effects on our calculations of scaling through a range of multiples $m\sigma_e$ for $m = 0, \ldots, 5$. The value $m = 0$ is chosen because of its conceptual interest - corresponding to the case of no discrepancy between the simulator output and the physical system - rather than as a realistic physical value.

7.6 Preliminary comparison with and without derivatives

We computed implausibilities for each output, using (7.4), over the 15^4 grid of corresponding active input points on which we adjusted beliefs in Chapter 6. Wherever we consider the effects of the number of runs on implausibilities, the ordering of runs is the same as in Chapter 6. For each input point in the grid, we combined implausibilities across outputs and stations based on (7.5), to produce $I_{\bar{P}}$, the maximum implausibility over the set, $\bar{P}$, of phytoplankton outputs across all four stations and, I_N, the maximum implausibility over the set, N, of mid-winter nutrient values at

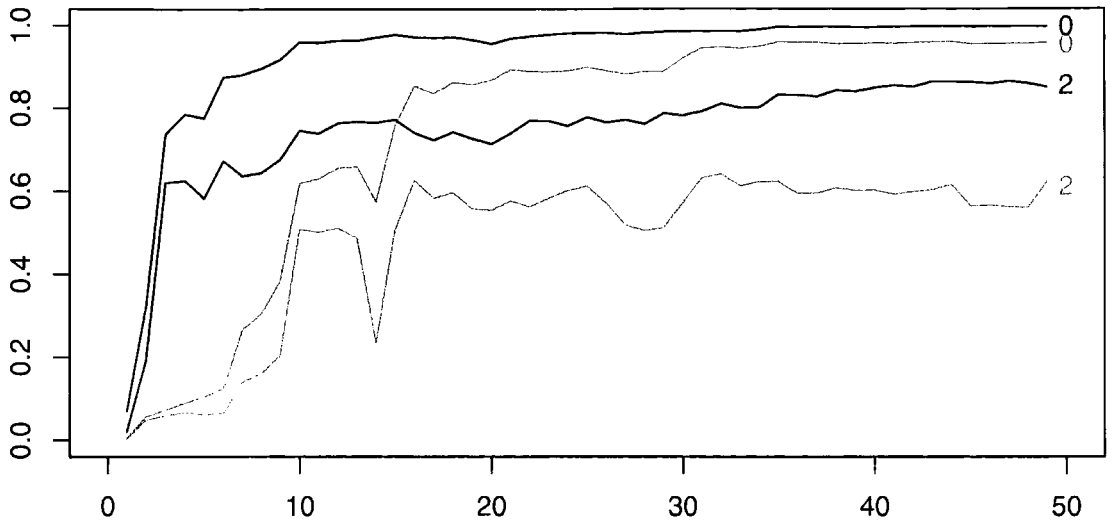


Figure 7.1: Proportion of input space ruled out by the set, $\bar{P}$, of phytoplankton outputs across all four stations with derivatives (black) and without derivatives (grey) plotted against the size n of the input set X_n at each station (so that e.g. $n = 10$ corresponds to ten runs at each station, and so forty in total). The numeric plotting symbols, $m = 0, 2$, correspond to $\sigma_\eta = m\sigma_e$, $m = 0, 2$.

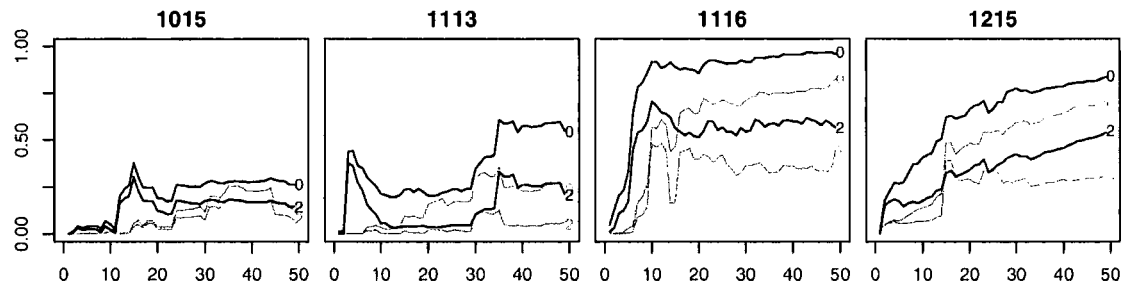


Figure 7.2: Proportion of input space ruled out by the set, $\bar{P}_s$, of phytoplankton output for each station, s, with derivatives (black) and without derivatives (grey).

the four stations. We did not combine these two measures at this stage in order to investigate the separate effects on calibration of the phytoplankton and nutrient outputs.

Figure 7.1 plots the proportion of input points ruled implausible based on $\bar{P}$ against the number of simulator runs, n, when derivatives are included (black) and

when they are not (grey) for the two cases $\sigma_\eta = 0$, $\sigma_\eta = 2\sigma_e$. For a given set of n input points, we see a marked improvement in our ability to rule out space when derivative information is included; with zero discrepancy, the proportion of input space ruled out after 5 runs at each station is 10.2% without derivatives compared to 77.6% with derivatives and, after 10 runs at each station, the corresponding proportions are 61.8% and 96%. For larger values of n, the difference begins to close as function values catch up, derivatives having effectively ruled out everything that is not plausible. For $\sigma_\eta = 2\sigma_e$, we see similarly large differences which prevail for large n.

Figure 7.3 plots the number of 'falsely' ruled points ruled out showing, for each n, the number of points which are currently ruled out which are ruled in based on $n = 50$ runs. The top left plot shows the effect at station 1116 which peaks at just over 10% of points after ten runs. The function values and derivatives for outputs at the tenth run are amongst the largest of those observed at station 1116 and this effect is apparently more localised than the emulator allows. The values are not particularly unusual and removing this observation does not register much of a change on the analysis. The top right hand plot shows the overriding cause; a slight sensitivity of the analysis to the cut-off value $c = 3$ above which we rule inputs as implausible. The top right hand plot shows the number of points that are ruled out after n runs based on a higher cutoff of $c = 4$ which are ruled in after 50 runs and we see that most of the effect has gone. There is still a peak after $n = 10$ runs but it quickly dissappears. In any case, we expect there to be a certain level of missclassification at each stage and are not too unhappy with the levels observed (See Craig et al. (1997, Section 8.1) for an interesting observation related to this). The bottom row shows the proportion of falsely ruled out points based on all four stations and here we see the benefits of having several stations, with the effect at station 1116 damped by their support. Overall the levels appear acceptable and, in general, appear to be lower when derivatives are included than when they are not.

The difference in our ability to rule out based on the set of nutrient outputs, N_w - shown in Figure 7.4 - is less marked, although it is still significant. For a given collection of σ_η values, derivatives consistently allow between 4% and 7% additional

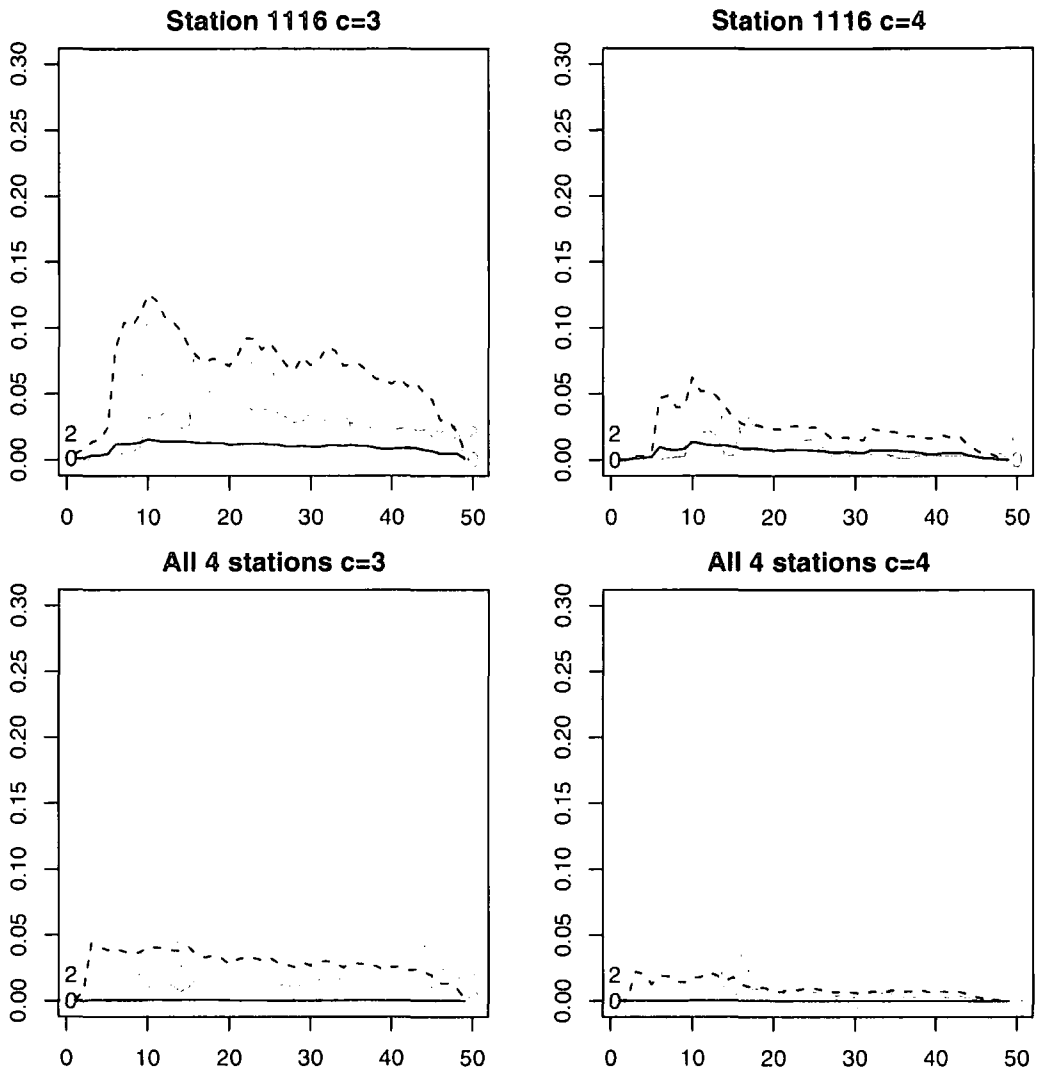


Figure 7.3: Proportion of points 'falsely' ruled out by phytoplankton outputs at station 1116 (top) and all four stations (bottom). The proportions are the number of points which are ruled out based on n input points but ruled in based on $n = 50$ points. Shown in each plot are the proportions with derivatives (black) and without derivatives (grey) for $\sigma_\eta = 0$ (thick line) and $\sigma_\eta = 2\sigma_e$ (dotted line). The cut off value in each plot is the implausibility score above which we rule points out.

space to be ruled out. Our ability to rule out a large amount of space before we make any runs is due to the strong linear effect of x_{13}, as respresented by a relatively large prior expectation for the β_{13} coefficient in the emulator for $N_{w,s}$, the form of

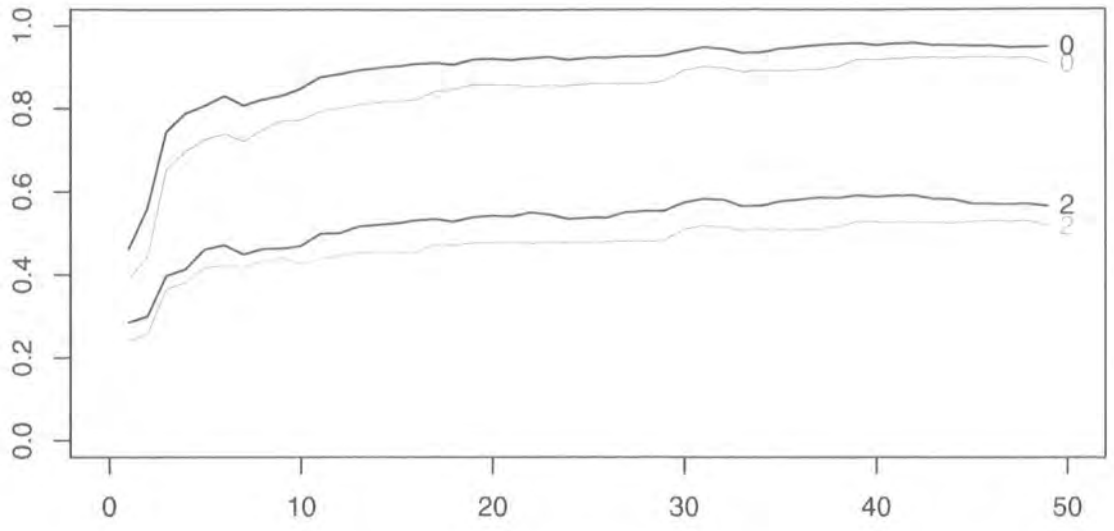


Figure 7.4: Proportion of input space ruled out by the set, N_w, of nutrient outputs across all four stations with derivatives (black) and without derivatives (grey). See Figure 7.4 for more details.

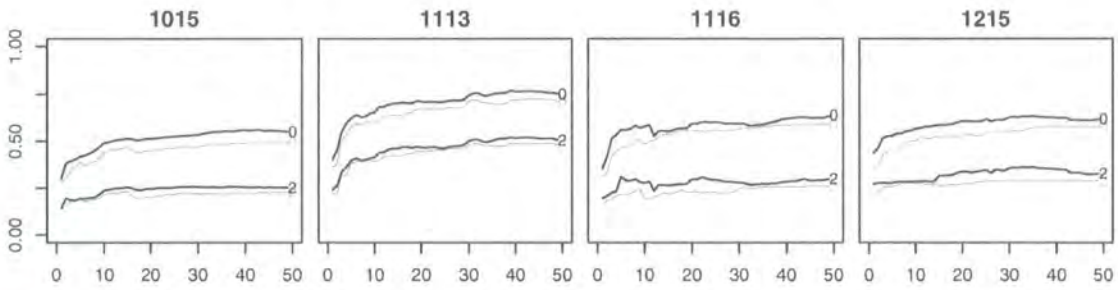


Figure 7.5: Proportion of input space ruled out by the nutrient output, $N_{w,s}$, at each station, s, with derivatives (black) and without derivatives (grey).

which is given in (6.5). Many of the points that are not ruled out *a priori* are ruled out by five runs or fewer because the variance relating to the x_{13} term is quickly resolved in the updating.

We see from the plots that, as we increase the number, n, of data points, we approach a 'saturation level' which is essentially an upper bound on the proportion of points we can rule out as implausible. As we get close to this level, any extra simulator data makes little difference to the calibration. The value of this saturation

depends on the collection of σ_η values and, in particular - as we would expect - the level decreases as we increase the scaling of the σ_η values. Close to this saturation level, reducing simulator variance further makes no difference because the change is small compared to the size of σ_η. For the $\bar{P}$, we have effectively reached this saturation level for the four stations combined after 10 runs with derivatives at each station whilst, we begin to get close at about $n = 30$ runs. Proportion of space ruled out by N_w and $\bar{P}$ for a few choices of n are given in Table 7.1. Note that, for calibration based on individual stations only (Figure 7.2), we are yet to reach a saturation level based on phytoplankton after $n = 50$ runs at each although we have for nutrient (Figure 7.5).

Note that although Figures 7.1 and 7.4 offer a natural way to compare calibration with and without derivatives for $\bar{P}$ and N_w individually, comparisons between the two groups using the plots may be somewhat ambiguous. This is because N_w has an active input - x_{13} - which is apparently much more active than the others so that, thinking of the proportion of ruled out points as a volume of the four dimensional active input space, we rule out based mainly on the x_{13} dimension and the resulting region is a 'wedge' which is thin in the x_{13} dimension and wide in the remaining active input dimensions. By contrast, the set $\bar{P}$ has no input that is dominant to the same extent, so that the ruling out of points is much more evenly spread through the active input dimensions.

As a final remark, it is worth noting that this increased 'speed' of learning about the emulator offers once again a lot of potential as a diagnostic - this time on our discrepancy variance: not only does derivative information allow us to rule out points more quickly, it allows us to see more quickly if we are ruling too much space out (in other words, when our discrepancy values are too low). This is important because we are concerned not to rule out points which currently have high-relative implausibility but which subsequently turn out to be good matches under the correct discrepancy specification. Such a scenario would be particularly disastrous if we had refocused our search in a sub-region of the input space having wrongly ruled out an alternative region because we had been too demanding of the simulator in our choice of discrepancy variance.

	N						$\bar{P}$					
m	0		1		2		0		1		2	
n	F	T	F	T	F	T	F	T	F	T	F	T
1	38.8	46.3	34.3	40.0	24.0	28.5	0.4	7.0	0.3	5.5	0.1	2.0
5	72.5	80.7	63.1	69.5	41.7	46.1	10.2	77.6	8.7	72.2	6.0	58.2
10	77.1	84.8	66.0	73.0	42.5	46.9	61.8	96.0	58.8	93.0	50.7	74.5
25	85.8	92.3	74.4	80.8	47.9	53.7	89.9	98.2	84.4	95.9	61.2	77.8
50	90.9	95.1	78.9	83.6	51.7	56.8	95.4	99.8	90.3	99.1	61.2	85.6

Table 7.1: Percentage of input points ruled implausible at each station based on n simulator runs and $\sigma_\eta = m\sigma_e$. F denotes the percentage when derivatives are not included and T when they are.

7.6.1 Function and derivative calibration trade-off

To conclude the comparison between function values and derivatives of calibration, we consider again horizontal and vertical gains as we did for emulator uncertainty reduction in Chapter 6. Since, in our discussion of falsely ruled out points, we concluded that levels were reasonably low, we compare trade-offs based on points ruled out. In any case, derivatives showed a slightly lower false rule out level in general and so consideration of this level would most likely benefit them. For the N_w, we have commented on the relatively small difference between function values and derivatives for calibration of the N_w. However, if we are concerned with questions of the type 'How many derivatives do I need to get a calibration as good as with fifty function values?' (horizontal difference), then the trade-off looks a lot more attractive; about ten. This is slightly misleading since ten function values allows us to rule out most of the same points. However, for phytoplankton outputs, a similarly impressive (if slightly better) horizontal trade-off gives rise to calibrations with which the same number of runs with only function values fails to compete. We would expect the trade-off to be favourable to derivatives, in general, for any outputs for which the ϵ-surface explains a significant proportion of the variation (that is, for outputs which are not too linear). For our simulator, the trade-off for

phytoplankton outputs of fifty function values versus ten runs with derivatives is a significant one and is strong in light of the increase in cost of calculating derivatives of only a factor of 2.

7.7 Calibration under maximal information

We now consider calibration using the full fifty runs at each station with derivatives in order to produce the best calibration for the physical problem. In all that follows, we take $\sigma_\eta = \sigma_e$ for all outputs. The exploration concludes with a discussion of our results in the light of the analysis by Hemmings et al. (2004).

7.7.1 Lower dimensional projections of implausibility

In order to visualise the plausible regions of the 4-dimensional input space, we produced plots of the lower dimensional projections of the implausibility surface by minimising over input dimensions using (7.6). Figure 7.6 shows 2-dimensional projections, $I_{\bar{P}}(x_1, x_4)$ (left) and $I_{\bar{P}}(x_6, x_{10})$ (right), with black dots denoting input points at which simulator output has been observed at one of the four stations and numbered circles marking the three most plausible input points in the 15^4 grid of evaluated points.

Broadly speaking, the first plot suggests a fairly distinctive most plausible region characterised by middle values of x_1 and mid-to high values for x_4. The plot of x_6 and x_{10} appears to suggest that combinations for which at least one of the two inputs is small offer the best matches, although again containing within this a distinctive most plausible region defined by low values of x_{10} and mid to low values of x_6. Both these inputs have strict physical lower bounds, $x_i = -1$ corresponding to zero in their original units, so there is no scope to stretch the search past these boundaries.

Figure 7.7 shows the same projections, but for implausibility maximised over the phytoplankton outputs at a single station only. We note that, as in Figure 7.6, we are able to rule more points out by the (x_1, x_4) projection than that for (x_6, x_{10}). We see also for both projections that, of the four stations, station 1116 allows us to rule out the most points.

It is worth adding a note of caution in the case of the (x_6, x_{10}) plot: the plots suggests a better model as x_{10} approaches its lower prior bound of $x_{10} = -1$. In such situations, where plausibility increases as we move towards a boundary, there is a chance that this may be the result of a lack of run data near to the boundary and, in particular, a lack of support of data from beyond the boundary leading to a higher uncertainty and in turn a low implausibility. In this case it is clear that there are plausible points near to the boundary (where the run data is) but the decreasing implausibility as we move away from them and towards the boundary may be the result of local decreasing linear slope imposed by implausible data above these points.

Figure 7.8 gives a 'four dimensional' representation of the implausibility surface. Outer grid lines correspond to the x_1 and x_4 dimensions with x_6 and x_{10} dimensions plotted, on the range $[-1, 1]$, within these outer grid lines. The most plausible 1% of points is shown, equating to just over 500 points in the 15^4 grid. From the two plots, we see that the majority of these most plausible points appear in the region 2 of the (x_1, x_4) space (the right-hand plot), defined by mid values of x_1 and high values of x_4. An interesting feature of this plot is that appears to suggest a higher dimensional interaction. In particular, the most plausible points in the lower part of this region tend to lie in region 2 of the (x_6, x_{10}) space whereas those closer to the boundary defined by $x_4 = 1$ tend to belong to region 1 of the (x_6, x_{10}) space. Such behaviour could have important implications for any future analysis, since it suggests that, for points in region 2 of the (x_6, x_{10}) space, there is no need to extend the search outside of the prior region past the prior upper bound for x_4.

Figure 7.9 shows this behaviour in action, with implausibility minimised only over the two most plausible regions (sub regions of regions 2 of both the (x_1, x_4) and (x_6, x_{10}) spaces. We see that the implausibility contours for the (x_1, x_4) space now move much more away from boundary of the x_4 interval, circling a region still in the upper part of the x_4 interval but reassuringly away from the boundary.

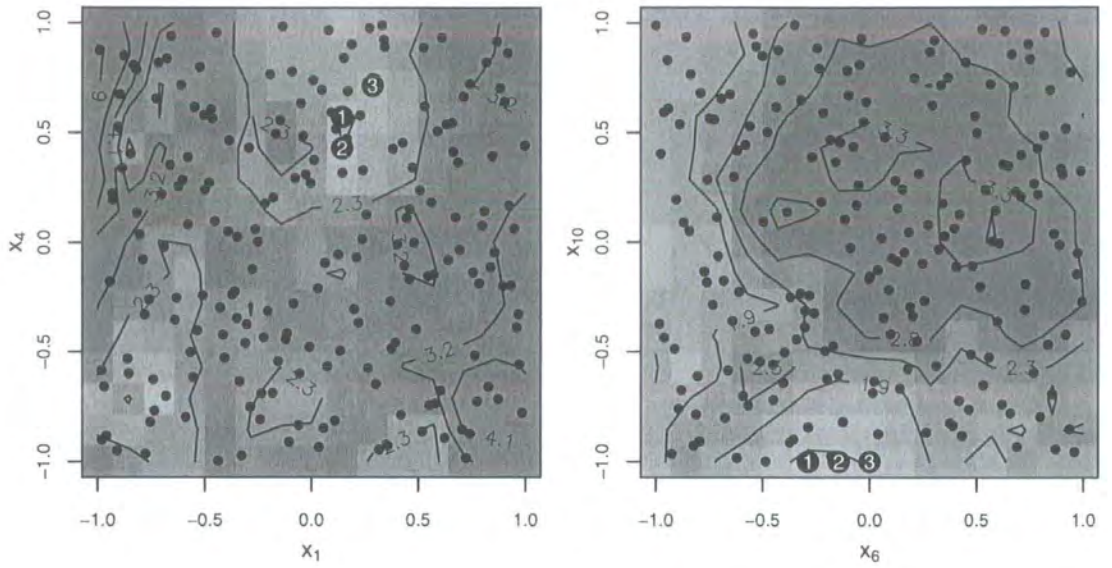


Figure 7.6: $I_{\bar{P}}(x_1, x_4)$ (left) and $I_{\bar{P}}(x_6, x_{10})$ (right), shown after $n = 50$ runs at each station. Implausible areas are shaded darker, black dots denote input points at which simulator output has been observed at one of the four stations, and numbered circles correspond to the three most plausible input points.

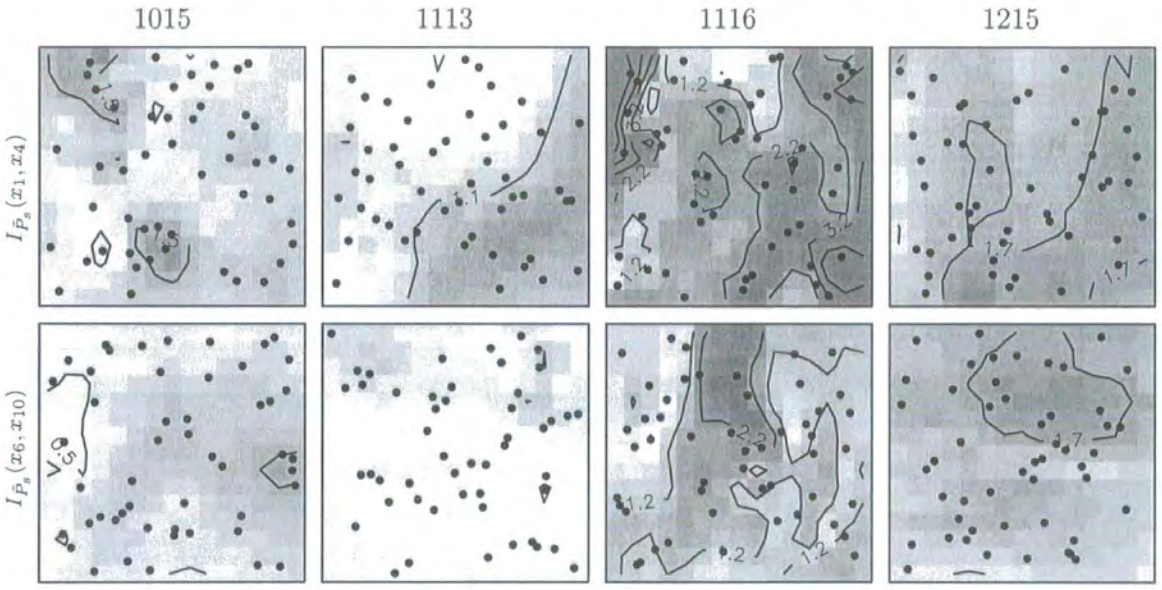


Figure 7.7: $I_{\bar{P}_s}(x_1, x_4)$ (top) and $I_{\bar{P}_s}(x_6, x_{10})$ (bottom): Implausibility maximised over phytoplankton outputs and projected into two input dimensions, for each of the four calibration stations, shown after $n = 50$ runs.

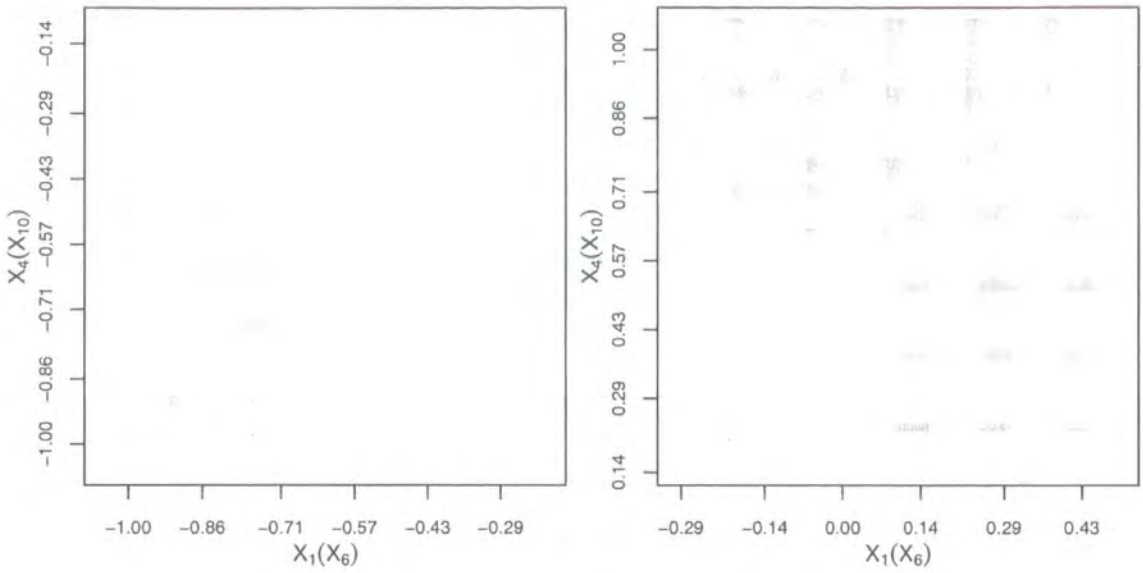


Figure 7.8: 'Four dimensional' plot of the top 1% 'most plausible' input combinations, all of which fall in either region 1 (left) or region 2 (right) of the (x_1, x_4) space. Outer grid lines correspond to the x_1 and x_4 dimensions with x_6 and x_{10} dimensions plotted within these grid lines, each on the range $[-1, 1]$.

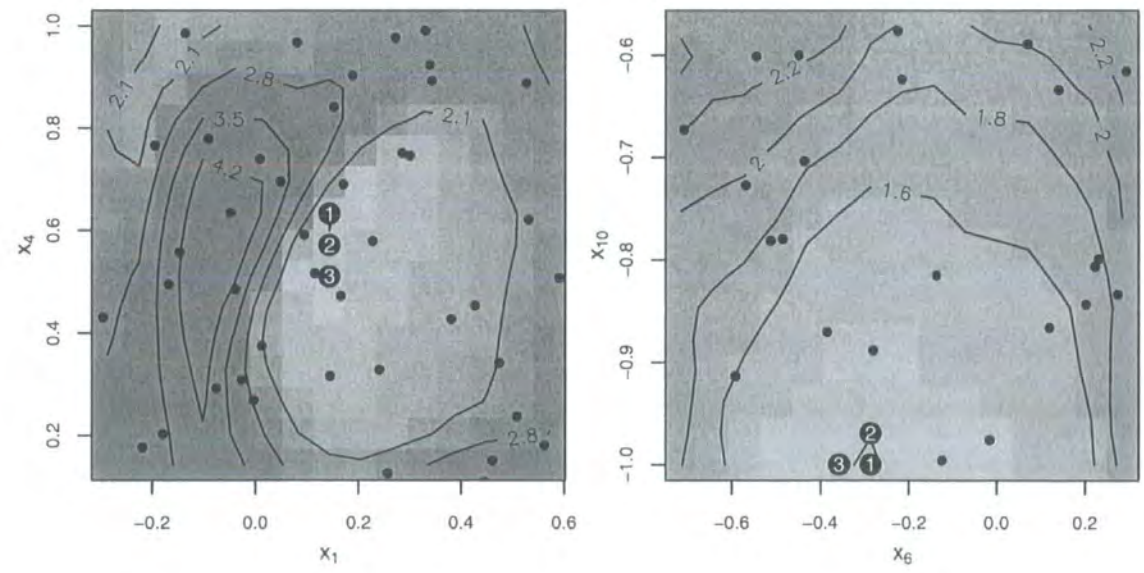


Figure 7.9: Subregion of two dimensional $I_{\bar{P}}$ projection in Figure 7.6, with implausibility minimised over the subregions only.

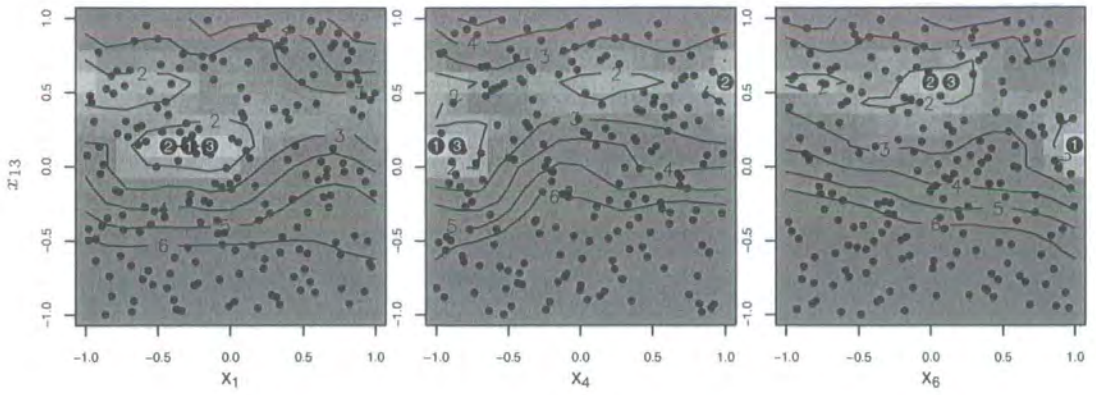


Figure 7.10: $I_N(x_i, x_{13})$ for $i = 1, 4, 6$: Implausibility maximised over nutrient outputs and stations and projected into two input dimensions by minimising over remaining input dimensions, shown after $n = 50$ runs at each station.

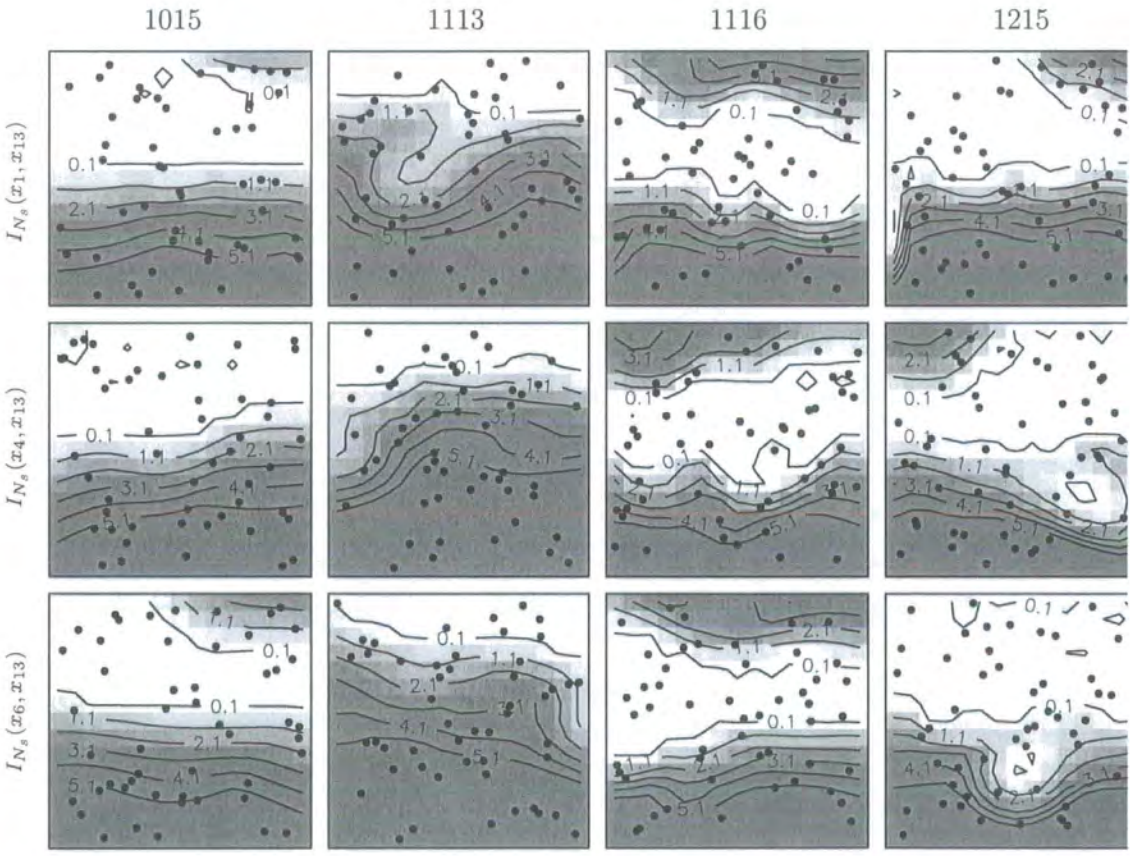


Figure 7.11: $I_{N_s}(x_i, x_{13})$ for $i = 1, 4, 6$: Implausibility based on the nutrient output at each of the calibration stations, projected into two input dimensions after $n = 50$ runs.

For nutrient outputs, the plots in Figure 7.7 demonstrate clearly the strong linear effect of x_{13} being felt in the calibration, with plausibility contours virtually orthogonal to the x_{13} axis in much of the space. Hence a side-effect of this is that, whilst N_w outputs would enable us to rule out a significant part of the x_{13} dimension, they would not offer us any help with the other input dimensions. The analogous plots for at the individual stations show a similar pattern and we see that we are able to rule out a significant part of the x_{13} dimension based on any one of the individual stations. The fact that there is so much overlap suggests that further calibration using nutrient values at other stations does not offer much scope for ruling out more space; as most of the information will be duplicate. Station 1113 looks at little bit different to the other three in the plausible regions. Based on the plots, we ruled out all input combinations outside the range $x_{13} \in [0, \frac{5}{7}]$, corresponding to $N_{\text{ref}} \in [9, 13.3]$ in the original units given in Table 2.1.

7.7.2 'Most plausible' output

In order to how see well the calibration was working, and to informally assess the magnitude of the discrepancy between the simulator and system, we ran the simulator at the four stations at $\hat{x}_0$, our best guess for x_0. In choosing a value for $\hat{x}_0$, we set all non-active inputs to be at their prior mean value. We choose components corresponding to active variables for $\bar{P}$ to be the point in the grid of implausibility evaluations which minimised $I_{\bar{P}}$, which lay in region 2 of both the (x_1, x_4) and (x_6, x_{10}) spaces. Note that this was a sensible choice in our case, because the final point had low uncertainty; in general, we want a best input with both small implausibility and small uncertainty. Finally, we took $x_{13} = 4/7$, corresponding to the midpoint of the plausible region based on the I_N plots. In theory, we could have minimised the latter over x_{13} and combined the resulting 3-dimensional grid with $I_{\bar{P}}$ when choosing values for x_1, x_4 and x_6, but we chose not to because of the ambiguity in how best to combine them and since, in any case, the result of the minimisation not surprisingly offered little constraining effect on these inputs. This gave, for the five active inputs, $(x_1, x_4, x_6, x_{10}, x_{13}) = \frac{1}{7}(1, 4, 0, -7, 4)$ in transformed units, corresponding to $(\phi_P, \alpha, k_P, g, N_{\text{ref}}) = (0.17, 0.16, 0.15, 0, 12.4)$ in the original

units given in Table 2.1.

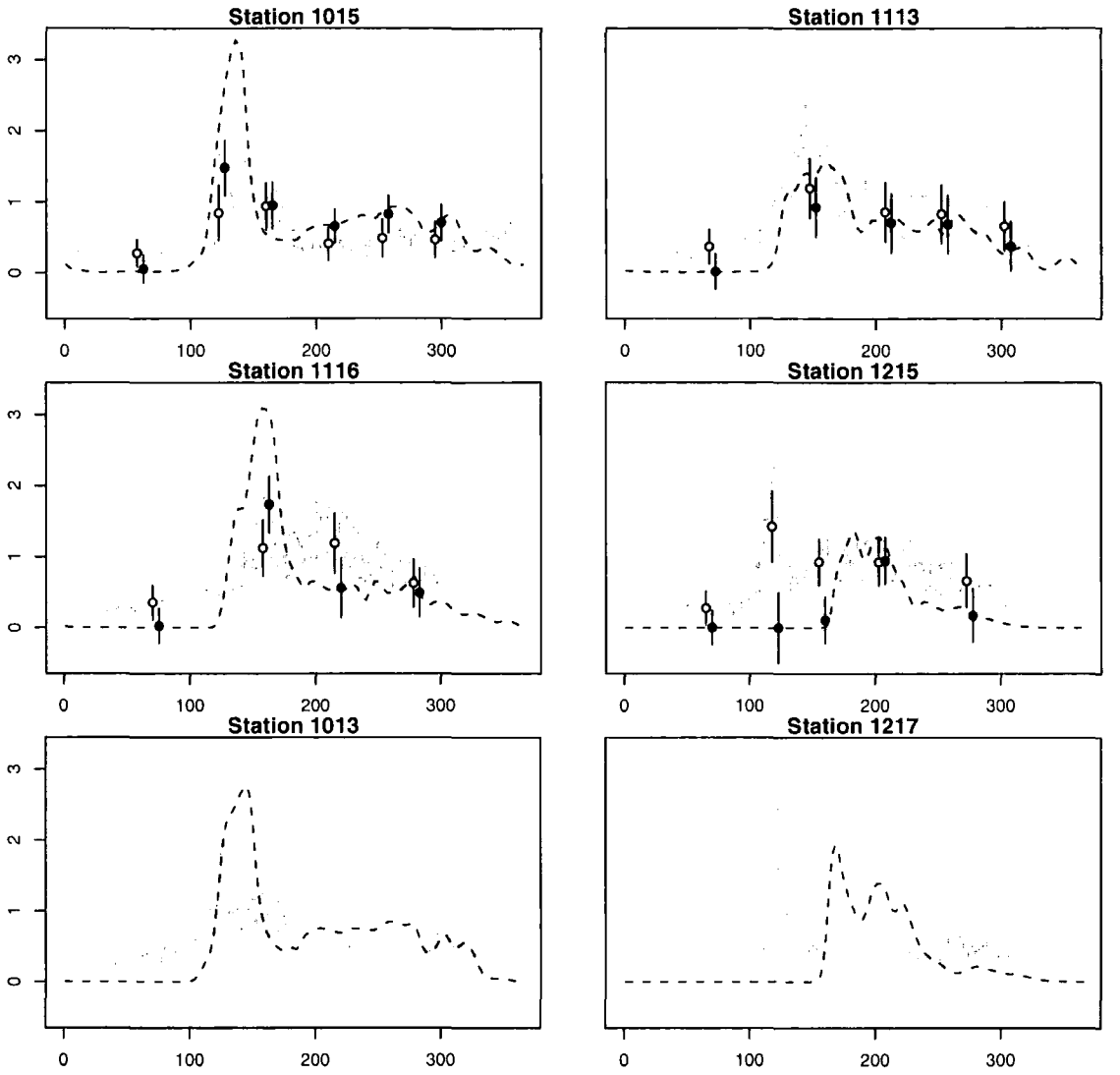


Figure 7.12: Simulator output at $\hat{x}_0$ (unfilled circles, with lines denoting $\pm 3\sigma_\eta$) and historical data (filled circles, with lines denoting $\pm 3\sigma_e$) for phytoplankton at the calibration stations. The dotted line gives the original, unparameterised phytoplankton output at $\hat{x}_0$ and the grey polygon shows the range spanned by $\pm 35\%$ of the original historical chlorophyll observations, as a guide to ± 3 measurement error for the calibration and validation stations.

Figure 7.12 shows the unparameterised phytoplankton output at the four calibration stations together with two further 'validation' stations, 1013 and 1217, lying at opposite ends of the calibration region. The grey polygon shows the range spanned by $\pm 35\%$ of the original historical chlorophyll observations, as a guide to

± 3 measurement error standard deviations, corresponding to the SeaWiFS target discussed in Section 7.5. The calibration station plots act as a visual diagnostic of our assertion, made through our choice of phytoplankton output parameterisation, that good matches on the interval mean values correspond broadly to good matches on the time series. The plots at the two validation station test the ability of the calibrated model to reproduce observations at uncalibrated stations. Of the 20 phytoplankton outputs across the calibration stations, all but two - the second and third outputs at station 1215 - match reasonably well.

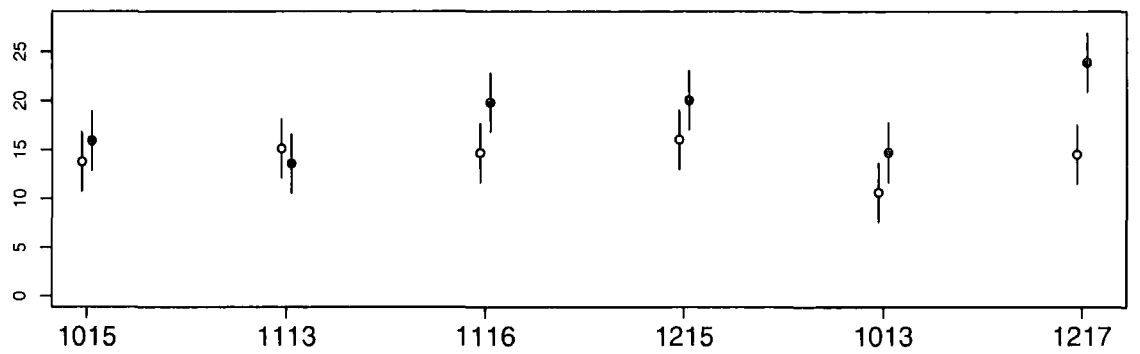


Figure 7.13: Simulator nutrient output at $\hat{x}_0$ (unfilled circles, with lines denoting $\pm 3\sigma_\eta$) and historical nutrient data (filled circles, with lines denoting $\pm 3\sigma_e$) at the calibration and validation stations. Validation stations are shown with $\sigma_e = \sigma_\eta = 1$ as a guide only.

Of the two validation stations, 1013 appears to be a reasonable match whilst 1217 looks to be a bad match. This is promising in our ability to reproduce output at station 1013 and also because 1217 offers us the potential to calibrate the inputs further at this level of output parameterisation. It is perhaps not surprising that we struggle at station 1217 since the closest station amongst the four calibration stations is 1215 and also station 1217 is one of those with a MLD forcing function that is different to many of the other stations (see Figure 6.1).

Figure 7.13 shows nutrient output at the six stations, with measurement error and discrepancy taken to be 1 at the validation stations - as it was at the calibration stations - as a guide. We see that nutrient outputs match well at the calibration stations with all four falling within ± 3s.d. of the historical observation. For the two

validation stations, we see that, as with phytoplankton, station 1013 looks to match well whilst 1217 once again appears more problematic.

s	$I_{\bar{P}_{s,1}}(\hat{x}_0)$	$I_{\bar{P}_{s,2}}(\hat{x}_0)$	$I_{\bar{P}_{s,3}}(\hat{x}_0)$	$I_{\bar{P}_{s,4}}(\hat{x}_0)$	$I_{\bar{P}_{s,5}}(\hat{x}_0)$	$I_{\bar{P}_{s,6}}(\hat{x}_0)$	$I_{N_s}(\hat{x}_0)$
1015	2.46	**3.21**	0.05	1.98	1.13	0.98	1.51
1113	**2.97**	1.21	0.76	0.57	1.69	-	1.08
1116	2.72	2.29	3.08	0.83	-	-	**3.59**
1215	2.36	**5.44**	2.65	0.11	2.68	-	2.83

Table 7.2: Implausibility scores at $\hat{x_0}$. Bold type denotes the output with maximum implausibility score at each station.

Table 7.2 gives the $I_{\bar{P}_s}(\hat{x_0})$ and $I_{N_s}(\hat{x_0})$ scores corresponding to the outputs in Figure 7.12, again taking $\sigma_\eta = \sigma_e$. The scores are computed using (7.4) with the simulator variance set to $\mathrm{Var}[s(x)] = \sigma_\delta^2$. We do not set the simulator variance to zero since the variation in the δ-surface, explained by the remaining inactive inputs, is not resolved by observing simulator output at $\hat{x}_0$. We do not expect changes in inactive inputs to make much difference but, since we have not optimised over them, it is likely that we will be able to find slightly improved fits by varying inactive inputs also.

We see that the largest value does indeed correspond to $\bar{P}_{1215,2}$, the bloom at station 1215 and that the matches at this station suggest that we may wish to scale up the discrepancy variance, at this station at least. Taking $\sigma_\eta = 2\sigma_e$ at station 1215 reduces all implausibility scores to below three except for $\bar{P}_{1215,2}$ which becomes 3.66.

As a quick way to offer insight into the quality of the match at $\hat{x}_0$, we computed the implausibility score of the best matching of the 50 simulator runs for each individual station, the results of which are shown in Table 7.3. It is not surprising that the runs do better in general than $\hat{x}_0$ since there are based on calibration at individual stations rather than across all four stations. As with $\hat{x}_0$, station 1215 is the most difficult to match locally: in the best matching of the simulator runs, we do better in the phytoplankton runs at 1215 at the expense of a poorly matching nutrient value, with an implausibility score of 5.45. In general it may not be so surprising that

$\bar{P}_{1215,2}$ is difficult to match since it is the narrowest of the parametrisation intervals across the four stations.

s	$I_{\bar{P}_{s,1}}(\hat{X}_0)$	$I_{\bar{P}_{s,2}}(\hat{X}_0)$	$I_{\bar{P}_{s,3}}(\hat{X}_0)$	$I_{\bar{P}_{s,4}}(\hat{X}_0)$	$I_{\bar{P}_{s,5}}(\hat{X}_0)$	$I_{\bar{P}_{s,6}}(\hat{X}_0)$	$I_{N_s}(\hat{X}_0)$
1015	2.84	1.49	1.18	0.93	1.49	1.12	0.74
1113	1.83	1.22	2.55	2.02	0.31	-	1.99
1116	2.31	2.64	2.56	2.19	-	-	1.17
1215	0.11	2.04	1.65	3.53	1.38	-	5.45

Table 7.3: Implausibility of input $\hat{X}_0$ corresponding to best matching set of output for each station from the fifty simulator runs.

Running the simulator at $\hat{x}_0$ gives us an idea of how well we can hope to match the simulator - we don't necessarily think this is the best we can do. We do however believe that the best run lies within the regions given in the previous section. We would refocus into these regions and, in this refocused domain, we might expect other inputs to become active. We might also parameterise the output into narrower intervals at this point, now that we have ruled out combinations which fail to reproduce our coarse parameterisation.

7.7.3 Refocusing and widening the scope of the calibration

A natural next step, given that we appear to be able to rule out a large part of the input space, is to *refocus*; that is to consider a sub-region of the input space, containing within it the input points which have not been ruled implausible and to repeat a similar analysis over this region. In particular, we should be able to build more accurate models of outputs over this reduced space and this may lead to new inputs becoming active (which can then be calibrated). We might also wish to re-parameterise outputs at this stage, for example, to be at a finer resolution.

The posterior input region (that is, the region we would continue to search were we to refocus) can be found as the direct product of the 2-dimensional implausibility projections (in fact this is an 'upper bound' on the plausible region since it ignores higher order effects, such as those shown in Figure 7.8, which would allow us to rule

out more space). Based on Figure 7.7, the posterior region for (x_1, x_4) is the union of region 1, defined by $[-1, -\frac{4}{7}] \otimes [-1, 0]$, and region 2, defined by $[-\frac{3}{7}, \frac{4}{7}] \otimes [\frac{1}{7}, 1]$. The posterior input region for (x_6, x_{10}) is the union of region 1, defined by $[-1, -\frac{4}{7}] \otimes [-\frac{4}{7}, 1]$, and region 2, defined by $[-1, 1] \otimes [-1, -\frac{4}{7}]$. The posterior region, then, is the direct product of these regions, together with $[0, \frac{5}{7}]$, the plausible interval for x_{13}.

The size of this posterior region can easily be computed as the product of the sizes of the lower dimensional regions. We have (as fractions of 15) for (x_1, x_4): $(4 \times 8) + (8 \times 7) = 88$; for (x_6, x_{10}): $15^2 - 11^2 = 104$; and for x_{13} : 6. Hence the posterior volume is $88 \times 104 \times 6 = 54,912$ which is approximately 7.2% of the original 15^5. The posterior volume, ignoring the x_{13} dimension, is approximately 18% of the prior input space. This compares to 91% of points ruled out after 50 runs at each station with derivatives in Table 7.1, representing a decrease of approximately 9 percentage points for our gain of easily parameterisable regions in which we can continue our search.

7.7.4 Discussion in relation to Hemmings analysis

Relating back to the work of Hemmings et al. (2004), the nature of the plausible regions in Figure 7.7 gives some insight into the difficulty Hemmings has in constraining individual parameters through his projections of joint posterior parameter distributions into marginal distributions for each input. In our two dimensional projections we see that, for each of the four active inputs for phytoplankton, the majority of the range for each input contains some plausible region. Interestingly, x_1 (ϕ_P in the original notation) - the one parameter that Hemmings is able to constrain well - would not be constrained well by a 1-dimensional projection of our 2-dimensional plot in Figure 7.7. Hemmings, who performs several calibration experiments over different groups of stations, finds all his best runs to be generated by $\phi_P \in [0, 0.03]$, corresponding to $x_1 \in$ [-1, -0.8] in our units. This would correspond with region 1 of the (x_1, x_4) plot in Figure 7.7. However, Hemmings' method may in fact be missing what appears to be the more favourable region 2 in our analysis because of his minimisation routine. Hemmings generates starting points to the

minimisation routine by adding scaled Normal(0,1) quantities to a prior value for each input where the scaling is done separately for the two sub-intervals either side of this prior value and is chosen to be a third of the length of each sub interval. For ϕ_P, his choice of prior value, at 0.05, is close to the lower end of the $[0, 0.3]$ which means that the minimisation routine is much more likely to start in the neighbourhood of the lower of our two regions and find its way there rather than find what our analysis suggests to be the best region (subject again to caveats about different implausibility measure, different output parametrisation etc.) In particular, we would argue that our region is preferable to that in Hemmings in view of our comments in Chapter 6 regarding Figure 6.8. We commented that the derivative of phytoplankton outputs with respect to x_1 appeared to be much more variable for small x_1. The consequence of this is that, whilst it is much more likely to be able to find good matches for low x_1, these matches are likely to be sensitive to small changes in x_1 and so, in particular, may not produce good matches at future uncalibrated stations. This may not be the case for all groups of calibration stations, however, Hemmings performs calibration experiments on several different groups of stations - although never exactly the group that we look at - and finds low ϕ_P are required in each; it may be that our new region is not valid for the groups that Hemmings considers.

Chapter 8

Conclusions and Future Directions

In this thesis, we have modified statistical methodology for computer models and applied it to a calibration problem of real-world interest. In setting the real-world problem in a rigorous statistical framework, we have obtained several useful insights of interest to people concerned with the specific model. In modifying the methodology, we have accomplished two main things. Firstly, we have made the first tentative steps in 'opening up the black box', exploiting the model structure to differentiate the simulator, re-closing the box and using the extra information generated to improve the process of belief building and updating of the emulator. Secondly, having offered a real-world example where derivatives are obtainable, we have provided what we believe to be the first serious investigation of the uses of derivatives in emulating computer simulators. In particular, we have shown that derivatives offer a range of natural ways to aid assessment of prior beliefs and that updating based on derivatives can lead to substantial reduction in emulator uncertainty. In addition, we have performed theoretical calculations, backed up by experimental results, of the cost of generating derivatives and considered the trade-off between derivatives and function values by comparing this cost against the associated reduction of uncertainties. We have shown that, for our model, the trade-off leans in favour of derivatives and discussed reasons as to why this will often be the case for compartmental models. Since compartmental models are widely applied in modelling physical systems, we believe that the applications of the work to be wide-spread. Moreover, we genuinely feel that anyone faced with a compartmental model of any complexity stands

to benefit a great deal by performing an analysis of the type developed in this thesis.

There are many possible future directions. Here are the most promising:

Multivariate covariance structure for output: Our study emulates each model output individually and does not consider covariance between outputs. In fact, we believe that taking outputs to be uncorrelated is a reasonable approximation for our coarse parameterisation. However, were we to re-parameterise phytoplankton output to be on a finer scale (e.g. following on from our work, after refocusing), then this analysis might benefit from a full multivariate covariance structure over outputs, although this would require a detailed and careful elicitation process. A more careful analysis and consideration of differences between forcing functions may be useful here. An advantage of this would be that multivariate implausibility measures would very naturally allow us to rule out input which not only produced one very bad output but, for example, obtained several 'OK but not great' matches in the output collection.

Epsilon covariance structure: Observing derivatives tells us about Θ and potentially allows a better estimate for its value than previously. Derivative information may assist us in improving Θ in two further ways. Firstly, as noted in Chapter 4, Θ can be generalised to be any positive definite matrix; it tends to be taken to be diagonal because it admits a convenient product form, but also because off-diagonal elements of Θ are even harder to estimate than those on the diagonal. It seems plausible, however, that just as derivatives have aided us in estimating the diagonal, they may also enable us to estimate the off-diagonal. Secondly, derivative information might also tell us that a constant Θ is not valid. This means that derivative information may be useful for other covariance functions; for example, a Gaussian-like covariance function with varying Θ. We don't know of such a covariance function and whether or not it would be positive definite, but it would be interesting to investigate. Alternatively, crudely breaking down input space into (a small number of) sub-regions, specifying different Θ in each, and then 'patching' the regions together might be a

way forward. Alternatively, transformations of input space, which 'warp' x, might be the answer so that the constant Θ assumption looks better.

Uncertainty on forcing functions: An interesting area for future work would be to investigate the effects of uncertainty about the model forcing functions. One way to proceed might be split the functions into a smoothed parameterised mean curve and residuals and then elicit uncertainty about the separate parts. In this way, we would effectively extend the number of input parameters and could, for example, differentiate with respect to these parameters.

Design: Derivatives allow us learn about the emulator based on less input points and provide information about regions of high local variability of output (that is, where derivatives are large). This might enable better-than-current sequential design where runs are targeted at these areas. Away from derivatives, we would recommend some control of orthogonality be exerted when using hypercube designs, as we discuss in Section 6.3.4. We give one problem-specific criterion - investigation into other criteria would be interesting.

Structure of compartmental models: The structure of compartmental equations, in general, appears to suggest that an input can't be active for an output without first being active for outputs that are connected directly to the flow function on which the input appears ('outputs to which the input is local'). We did not pursue this because, with only three compartments, any such effects transferred through our model very quickly. However, for a much larger model with hundreds of compartments, it may be useful to make use of this structure when emulating outputs; for example, in choosing active inputs. In addition, linking this to our comments about the ϵ covariance structure, consideration of this structure may lead to us considering a block diagonal structure for Θ with blocks corresponding to groups of parameters which are local to each other.

Bibliography

Brock, T. D. (1981). Calculating solar radiation for ecological studies. *Ecological Modelling 14*, 1 – 14.

Buck, C., W. Cavanagh, and C. Litton (Eds.) (1996). *Bayesian Approach to Interpreting Archaeological Data.* Chichester: John Wiley and Sons.

Burden, R. L. and J. D. Faires (1989). *Numerical Analysis Fourth Edition.* Boston: PWS-KENT.

Conkright, M., T. O'Brien, S. Levitus, T. Boyer, J. Antonov, and C. Stephens (1998). World ocean atlas 1998 vol 10: Nutrients and chlorophyll of the atlantic ocean. noaa atlas nesdis 36. Technical report, US Govt. Printing Ofice, Washington.

Cooke, R. and B. Kraan (2000a). Processing expert judgements in accident consequence modelling. *Radiation Protection Dosimetry 90*(3), 311 – 315.

Cooke, R. and B. Kraan (2000b). Uncertainty in compartmental models for hazardous materials - a case study. *Journal of Hazardous Materials 71*, 253 – 268.

Cowell, R., A. David, S. Lauritzen, and D. Spiegelhalter (1999). *Probabilistic Networks and Expert Systems.* New York: Springer.

Craig, P. S., M. Goldstein, J. C. Rougier, and A. H. Seheult (2001). Bayesian forecasting for complex systems using computer simulators. *Journal of American Statistical Association 96*, 717 – 729.

Craig, P. S., M. Goldstein, A. H. Seheult, and J. A. Smith (1996). Bayes linear strategies for history matching of hydrocarbon reservoirs (with discussion). *Bayesian Statistics 5*, 69 – 95.

Craig, P. S., M. Goldstein, A. H. Seheult, and J. A. Smith (1997). Pressure matching for hydrocarbon reservoirs: A case study in the use of bayes linear strategies for large computer experiments (with discussion). *Case Studies in Bayesian Statistics III*, 36 – 93.

Craig, P. S., M. Goldstein, A. H. Seheult, and J. A. Smith (1998). Constructing partial prior specifications for models of complex physical systems (with discussion). *Journal of Royal Statistical Society 47*, 37 – 53.

Currin, C., T. Mitchell, M. Morris, and D. Ylvisaker (1991). Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments. *Journal of the American Statistical Association 86*, 953 – 963.

de Finetti, B. (1974). *Theory of Probability*, Volume 1. London: Wiley.

Dimitrios, M., I. Kioutsiouskis, I. Zerefos, and I. Ziomas (2003). Evolutions of perturbations in 3d air quality models. *Annals of Geophysics 46*(2), 353 – 361.

Goldstein, M. (1999). Bayes linear analysis. In S. Kotz, C. Read, and D. Banks (Eds.), *Encyclopedia of Statistical Sciences*. New York: Wiley. Update Volume 3.

Goldstein, M. and J. C. Rougier (2004). Probabilistic formulations for transferring inferences from mathematical models to physical systems. *SIAM Journal on Scientific Computing, forthcoming*.

Gordon, H. (1993). Radioactive transfer in the atmosphere for correction of ocean colour remote sensors. In V. Barale and P. Schlittenhardt (Eds.), *Ocean Colour: Theory and applications in a Decade of CZCS Experience*. ECSC,EEEC,EAEC.

Gradstein, I. S. and I. M. Ryshik (1981). *Tables of Series, Products and Integrals Volume 2*. Frankfurt: Verlag Harri Deutsch.

Graybill, F. A. (1983). *Matrices with Applications in Statistics: Second Edition.* California: Wadsworth.

Griewank, A. (1989). On automatic differentiation. In M. Iri and K. Tanabe (Eds.), *Mathematical Programming: Recent Developments and Applications*, pp. 83 – 107. Tokoyo: KTK Scientific Publishers.

Hemmings, J. (2000). Validation of oceanic ecosystem models using satellite ocean colour and data assimilation techniques: a preliminary experiment. Technical Report 58, Southampton Oceanography Centre.

Hemmings, J. C. P., M. A. Srokosz, P. Challenor, and M. J. R. Fasham (2003). Assimilating satellite ocean colour observations into oceanic ecosystem models. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical, Physical and Engineering Sciences 361*, 33 – 39.

Hemmings, J. C. P., M. A. Srokosz, P. Challenor, and M. J. R. Fasham (2004). Split-domain calibration of an ecosystem model using satellite ocean colour data. *Journal of Marine Systems 50*, 141–179.

Jia, Y. (2000). Formation of an azores current due to mediterranean overflow in a modelling study of the north atlantic. *Journal of Physical Oceanography 30*, 2342 – 2358.

Kennedy, M. and A. O'Hagan (2001a). Bayesian calibration of computer models (with discussion). *Journal of the Royal Statistical Society, Series B 63*(3), 425–464.

Kennedy, M. and A. O'Hagan (2001b). Supplementary details on bayesian calibration of computer models. Technical report, University of Sheffield.

Kennedy, M. C. and A. O'Hagan (2000). Predicting the output from a complex computer code. *Biometrika 87*, 1 – 13.

Koehler and Owen (1996). Computer experiments. In S. Ghosh and C. Rao (Eds.), *Handbook of Statistics*, Volume 13, pp. 261–308. Elsevier.

Levitus, S. (1982). Climatological atlas of the world ocean. Technical Report 13, US Govt. Printing Ofice, Washington.

McClain, C. R., M. L. Cleave, G. C. Feldman, W. W. Gregg, S. B. Hooker, and N. Kuring (1998). Science quality seawifs data for global biogeochemical research. *Sea Technology 39*, 10 – 16.

Mitchell, T., M. Morris, and D. Ylvisaker (1993). Bayesian design and analysis of computer experiments: Use of derivatives in surface prediction. *Technometrics 35*, 243 – 255.

Mitchell, T., M. Morris, and D. Ylvisaker (1994). Asymptotically optimum experimental designs for prediction of deterministic functions given derivative information. *Journal of Statistical Planning and Inference 41*, 377 – 389.

Näther, W. and J. Šimák (2003). Effective observation of random processes using derivatives. *Mathematical Programming: Recent Developments and Applications 58*, 71 – 84.

Oakley, J. E. (2002). Eliciting gaussian process priors for complex computer codes. *The Statistician 51*, 81 – 97.

Oblow, E. M., F. Pin, and R. Q. Wright (1986). Sensitivity analysis using computer calculus: A nuclear waste isolation application. *Nuclear Science and Engineering 94*, 46 – 65.

O'Hagan, A. (1978). Curve fitting and optimal design for prediction (with discussion). *Journal of the Royal Statistical Society Series B 40*, 1 – 42.

O'Hagan, A. (1992). Some bayesian numerical analysis (with discussion). In J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. Smith (Eds.), *Bayesian Statistics*, Volume 4, Oxford, U.K., pp. 345 – 363.

O'Hagan, A. (1998). A markov property for covariance structures.

O'Reilly, J. E., S. Maritorena, and M. O'Brien (2000). Sea wifs postlaunch calibration and validation analyses. Technical Report 3, NASA.

Owen, A. B. (1992). A central limit theorem for latin hypercube sampling. *Journal of the Royal Statistical Society B 54*, 541 – 551.

Pawitan, Y. (2001). *In all likelihood: Statistical modelling and inference using likelihood.* Oxford: Oxford University Press.

Press, W., B. Flannery, S. Teukolsky, and W. Vetterling (1992). *Numerical Recipes in C, 2nd edition.* Cambridge: Cambridge University Press.

Pukelsheim, F. (1994). The three sigma rule. *The American Statistician 48*, 88–91.

R Development Core Team (2004). *R: A language and environment for statistical computing.* Vienna, Austria: R Foundation for Statistical Computing. ISBN 3-900051-00-3.

Ripley, B. D. (1988). *Statistical processes for spatial Processes.* Cambridge: University Press.

Rougier, J. C. (2004). Bayesian prediction of climate using ensembles of simulator evaluations. Submitted to *Climate Change.*

Sacks, J., W. J. Welch, T. J. Mitchell, and H. P. Wynn (1989). Design and analysis of computer experiments (with discussion). *Statistical Science 4*, 409 – 435.

Santner, T. J., B. J. Williams, and W. I. Notz (2003). *The Design and Analysis of Computer Experiments.* New York: Springer-Verlag.

Schlather (1999). Introduction to positive definite functions and to unconditional simulation of random fields. Technical Report ST-99-10, Lancaster University.

Stein, M. L. (1999). *Interpolation of spatial data.* New York: Springer-Verlag.

Yaglom, A. M. (1986). *Correlation Theory of Stationary and Related Random Functions*, Volume I. New York: Springer-Verlag.

Appendix A

Additional Chapter 6 analysis

This section contains derivative and diagnostic plots for other stations not included in Chapter 6.

A.1 Derivative plots

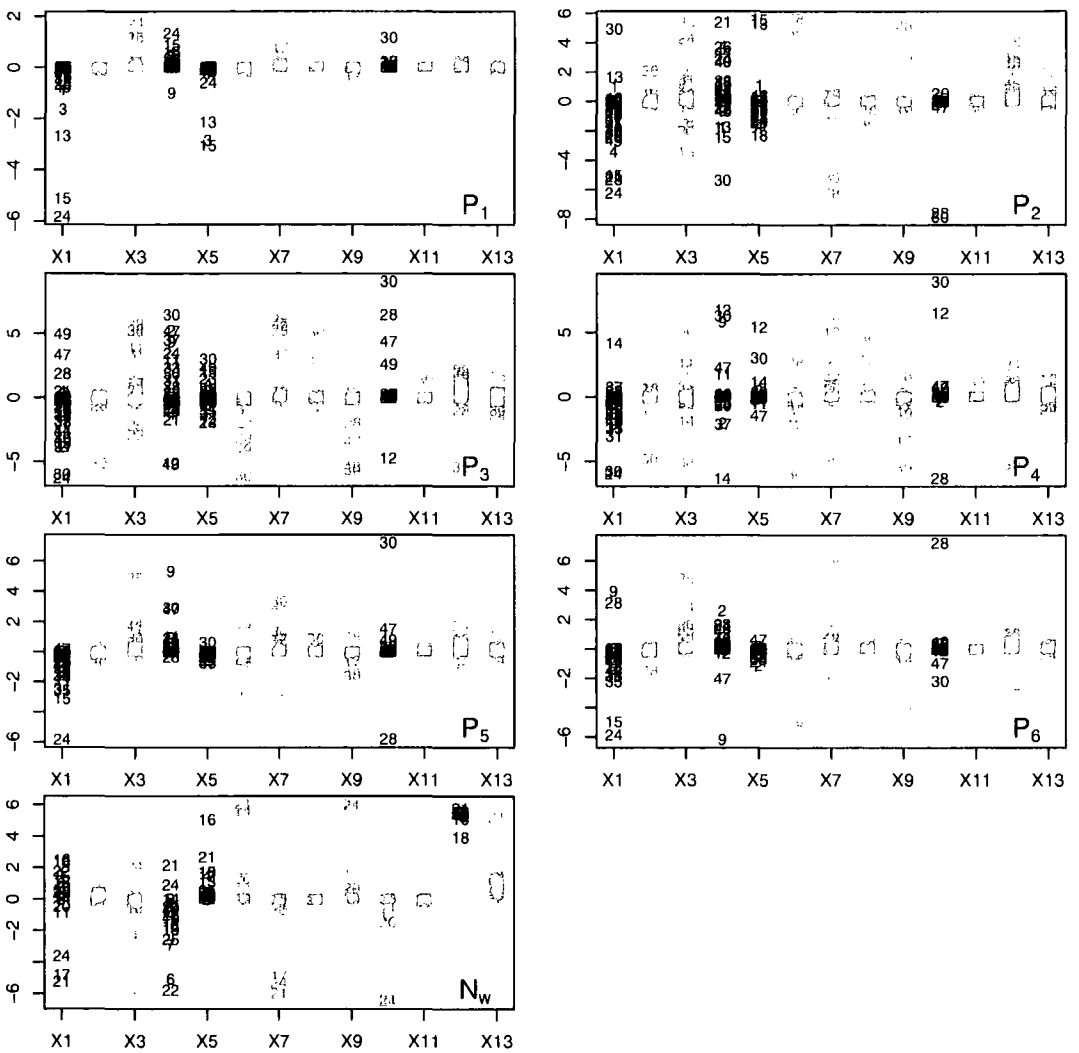


Figure A.1: Station 1015

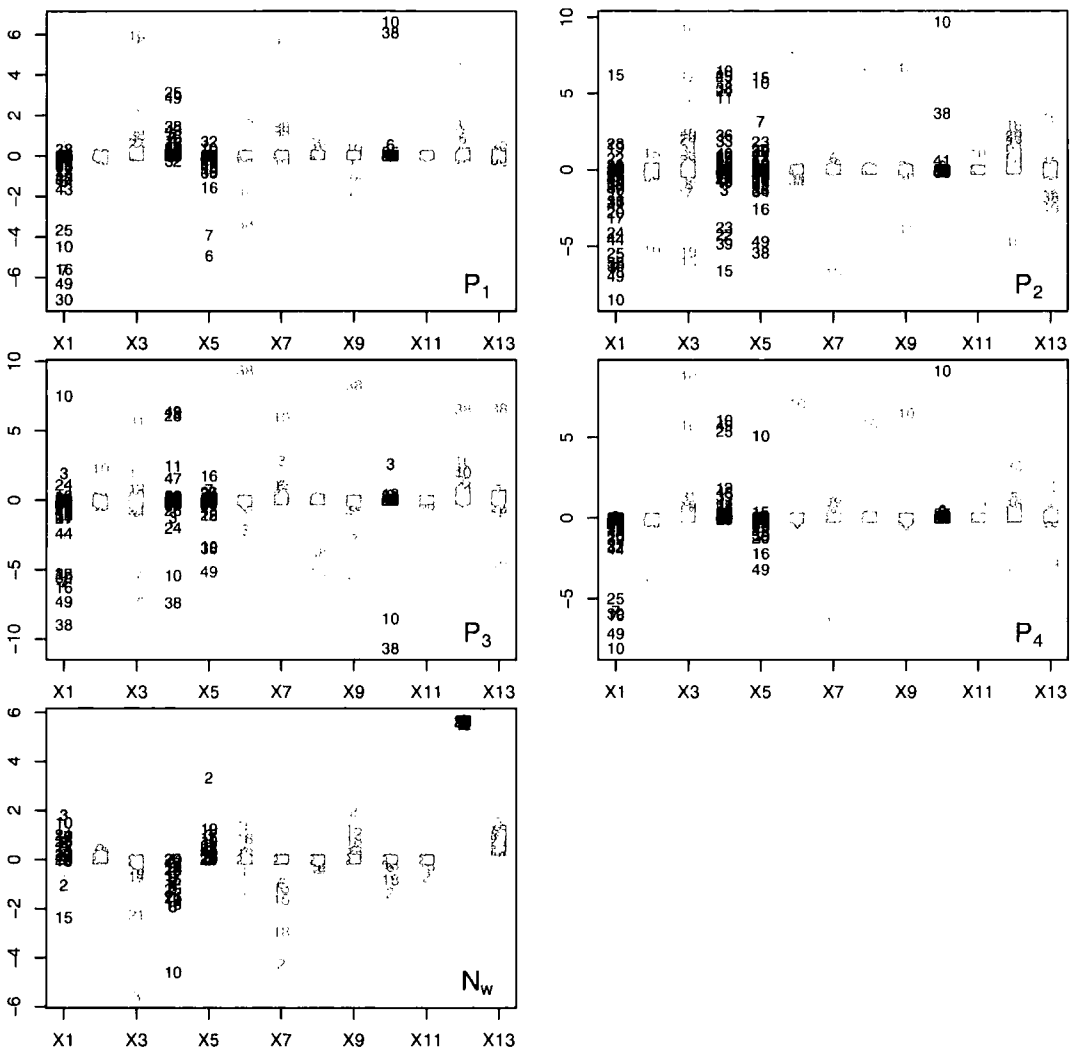


Figure A.2: Station 1116

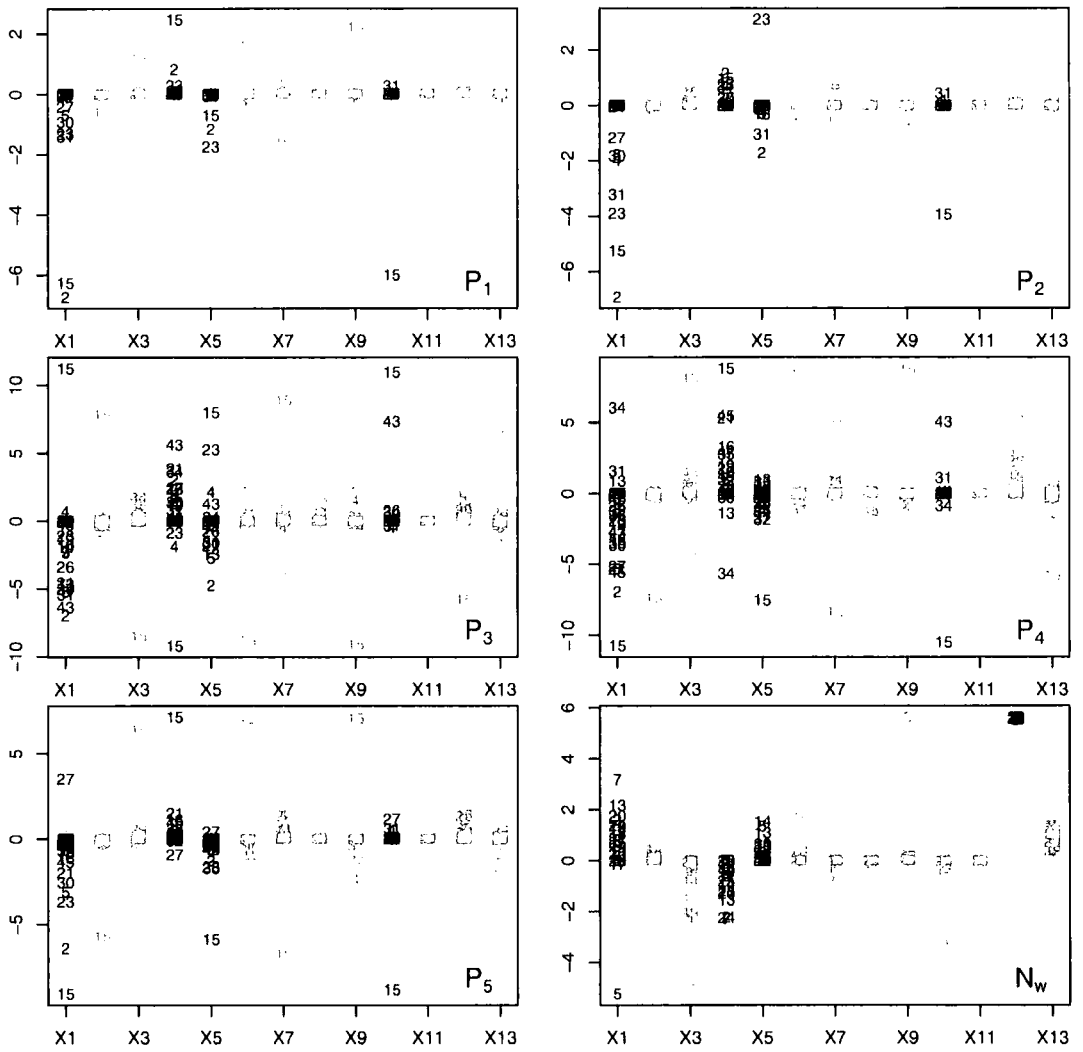


Figure A.3: Station 1215

A.2 Simulator Diagnostics

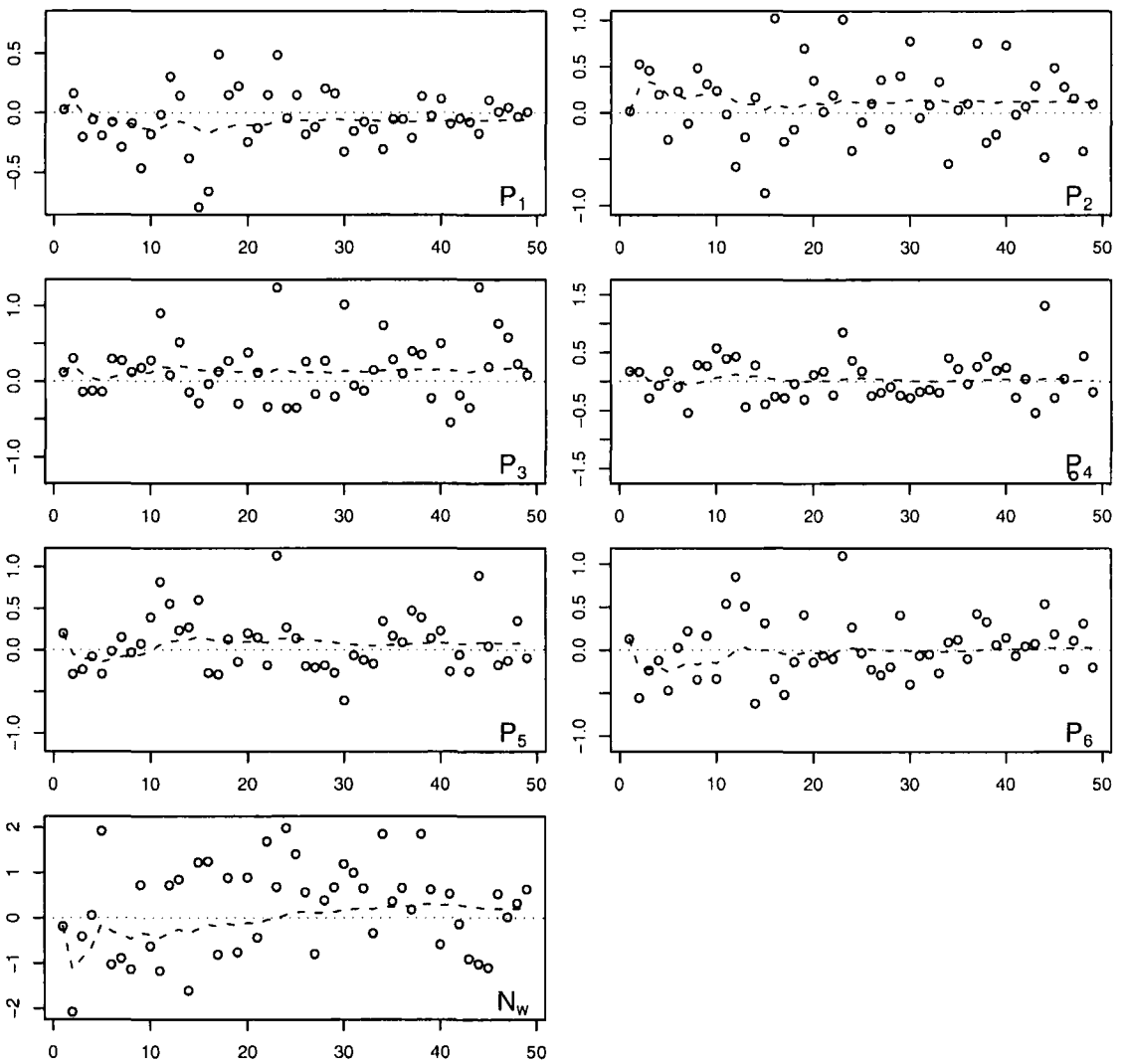


Figure A.4: Station 1015

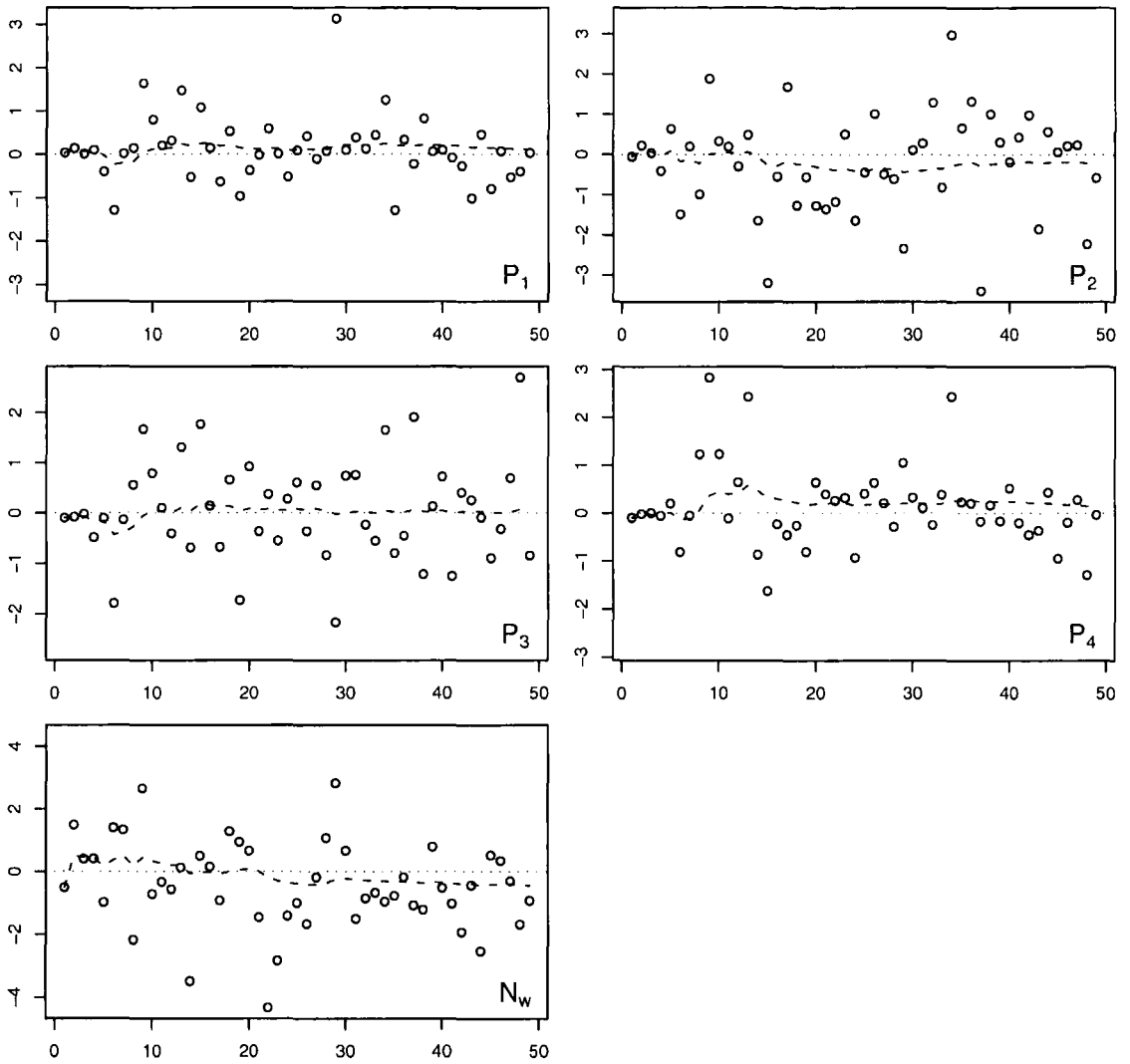


Figure A.5: Station 1116

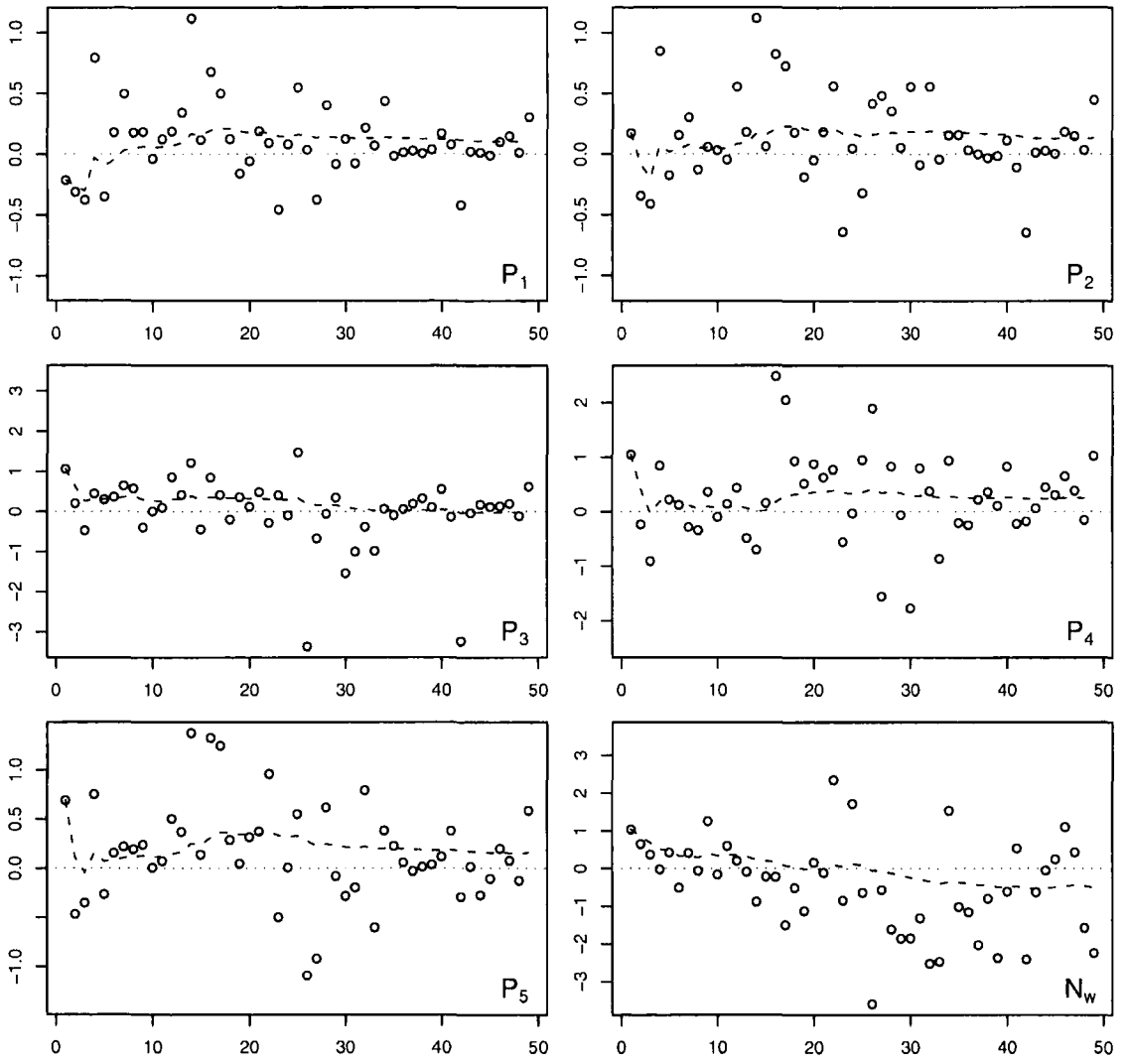


Figure A.6: Station 1215

Appendix B

PZN simulator run code with derivatives

```
Solve.PZN<- function( station, runnames, deriv.order, act=1:15,
                      runnamesX=runnames, adapt="all"){

  pars<-c("phiP", "kN", "VP", "alpha", "kW", "kP", "beta", "mu", "phiZ",
          "g", "kG", "epsilon", "Nref", "b", "m")

  p<-length(pars)
  NULLVEC<-rep(0, p)
  pact<-length(act)

  if (deriv.order==0){
    NAM<-c( "per", "tdays", "P", "Z", "N" )
  } else {
    NAM<-c( "per", "tdays", "P", paste("dPd", pars, sep=""),
          "Z", paste("dZd", pars, sep=""), "N", paste("dNd", pars, sep="") )
    RANK<-rep(NA, p)
    RANK[act]<-rank(act)
  }


  inp<-read.table( paste(runnamesX,".X",sep=""), header=TRUE  )
  # Given in original units

  cd<-ncol(inp)
  if (!cd==p) {
```

```
    stop("Only", cd, "parms given - should be", p)
  }

  nperiod<-2 # One year's spin-up as in Hemmings

# Read in ffs
if (any(station==10010+1:4)){
  ffpath<-file.path( "~dma3mrk", "Hemmings", "PrelimData" )
  lats<-read.table( file.path(ffpath, "stations.nabe" ), header=T )
  CLOUD<-read.table( file.path(ffpath, "cloud.nabe"), header=T )
  MLD<-read.table( file.path(ffpath, "mld.nabe"), header=T )
} else {
  lats<-as.matrix( read.table(file.path("~/Hemmings","stations.5d"), head=T) )
  CLOUD<-as.matrix( read.table(file.path("~/Hemmings","ffcloud"), head=T) )
  MLD<-as.matrix( read.table(file.path("~/Hemmings","ffmld"), head=T) )
}

  lat<-lats[which(lats[,"station" ]==station), "lat" ]
  lat<-lat*pi/180 # convert into radians

  CLOUD<-CLOUD[ which(CLOUD[,"station"]==station), c("tdays","cloud") ]
  MLD<-MLD[ which(MLD[,"station"]==station), c("tdays","mld") ]
  CLOUD<-as.matrix(CLOUD)
  MLD<-as.matrix(MLD)

  nrowM<<-nrow(MLD)
  nrowC<<-nrow(CLOUD)

  CLOUD<-rbind( c( CLOUD[nrowC,1,drop=F]-365, CLOUD[nrowC,2,drop=F] ), CLOUD,
                c( 365+CLOUD[1,1,drop=F], CLOUD[1,2,drop=F] )  )
  MLD<-rbind(   c( MLD[nrowM,1,drop=F]-365, MLD[nrowM,2,drop=F] ), MLD,
                c( 365+MLD[1,1,drop=F], MLD[1,2,drop=F] )  )


getff<-function(tdays){


# CLOUD

while ( CLOUD[Ccur,1]>tdays ) { Ccur<<-Ccur-1 }
```

```
while ( all(CLOUD[ min(nrowC,Ccur+1), 1]<=tdays, Ccur<nrowC) ) {
  Ccur<<-Ccur+1
}

if (CLOUD[Ccur,1]==tdays) {
  C<-CLOUD[Ccur,2]
} else {
  w1<-CLOUD[Ccur+1,1]-tdays; w2<-tdays-CLOUD[Ccur,1]; w<-w1+w2;
  C<-( CLOUD[Ccur,2]*w1 + CLOUD[Ccur+1,2]*w2 )/w
}


# MLD

while ( MLD[Mcur,1]>tdays ){ Mcur<<-Mcur-1 }
# Careful with bad jumps as Mcur is set globally

while ( all( MLD[ min(nrowM,Mcur+1), 1]<=tdays, Mcur<nrowM) ){
  Mcur<<-Mcur+1
}

 if (MLD[Mcur,1]==tdays) {
  M<-MLD[Mcur,2]
} else {

   w1<-MLD[Mcur+1,1]-tdays; w2<-tdays-MLD[Mcur,1]
   if (any(w1<0, w2<0)){
    cat( paste( "M: t=", round(tdays,3), "w1=", round(w1,3),
                "w2=", round(w2,3), "Mcur=",Mcur), "\n" )
        w1<-abs(w1); w2<-abs(w2)
      }
      w<-w1+w2
      M<-( MLD[Mcur,2]*w1 + MLD[Mcur+1,2]*w2 )/w
}


      w1<-MLD[Mcur,1]-MLD[Mcur-1,1]; w2<-MLD[Mcur+1,1]-MLD[Mcur,1]
      if (any(w1<0, w2<0)){
        print( cat( "dMdt: t=", round(tdays,3), "w1=", round(w1,3),
                    "w2=", round(w2,3), "Mcur=",Mcur),"\n" )
        w1<-abs(w1); w2<-abs(w2)
```

```
        }
        w<-w1+w2
        m1<-MLD[Mcur+1,2]-MLD[Mcur,2]; m2<-MLD[Mcur,2]-MLD[Mcur-1,2];

    dMLDdt<- ( w1*m1/w2 + w2*m2/w1 )/w

    c(C, M, dMLDdt)

}


# The two Phot functions
# Removed lny (denom of ln) - subtract this later
    Ffn<-function(y,t){

        s2<- y^2 + t^2
        s<-sqrt(s2)
        s - t*log( (t + s)/y )

    }

    dFfn<-function(y,t){

        s2<- y^2 + t^2
        s<-sqrt(s2)
        (s2 + t*s)/( y*(t+s) )

     }

   phint<-function(y){

     y * (0.555588 + 0.004926*y) / (1 + 0.188721*y)

   }

  #Let y->1/360*y and mutiply top and bottom by (360y)^2

   phint2<-function(arg){

         num<- (61732*arg + 1.52037037)
         den<- arg*( arg*0.4e8 + 20969 )
```

```
        num/den

  }

  dphint<-function(arg){

     num<- - ( 0.246928e13 + (0.1216296296e9/arg) + (0.3188064629e5/arg^2)  )
     den<- ( arg*0.4e8 + 20969 )^2

     num/den

  }


# RK constants

  Ca<-c( NA, 1/5, 3/10, 3/5, 1, 7/8 )
  Cb<-matrix( NA, nrow=6, ncol=5)
  Cb[2,1]<- 1/5; Cb[3,1:2]<- c(3/40, 9/40)
  Cb[4,1:3]<- c(3/10, -9/10, 6/5)
  Cb[5,1:4]<- c(-11/54, 5/2, -70/27, 35/27)
  Cb[6,1:5]<- c(1631/55296, 175/512, 575/13824, 44275/110592, 253/4096)

  Cc<-c( 37/378, 0, 250/621, 125/594, 0, 512/1771 )
  dCc<-Cc-c( 2825/27648, 0, 18575/48384, 13525/55296, 277/14336, 1/4 )

  SAFETY<- 0.9
  PGROW<- -0.2
  PSHRINK<- -0.25
  ERRCON<- 1.89e-4 #ERRCON=(5/SAFETY)^(1/PGROW)
  HMIN<- 1e-50
  TINY<- 1e-25 #1e-30 seems too demanding
  EPS<- 1e-4 # form 1e-6

  HINITIAL<- 1
  HMAX<<- 25 # form 5
  actualHMAX<<-HMAX
  MAXnsteps<- 1e4 # counts two periods together
  NSTEPS<-c()
```

```
 fun<- function(tdays, Ymat, deriv.order=1){

   P<-Ymat[1,1]; Z<-Ymat[2,1]; N<-Ymat[3,1]

   ffs<-getff(tdays); C<-ffs[1]; M<-ffs[2]; dMdt<-ffs[3]

   if (M<0){ print( paste("M<0 =",M, "t=", tdays) ) }

   delta<- -0.40928*cos( 2*pi*cos( (tdays+10)*(2*pi/365) )/365 )


   ss<-sin(delta)*sin(lat)
   cc<-cos(delta)*cos(lat)
   tt<- -ss/cc

   if (tt <= -1) {
     sunset<-pi
   } else {

     if (tt >= 1) {
       sunset<-0
     } else {
       sunset<-acos(tt)
     }
   }
   #Sunset is an angle (which R gives in Radians)

   tau<-sunset/360

#  884*pi = 2777 - NB 1353 in Hemmings(2001) documentation; 430*pi = 1353
   Isn<- 884*( ss*sunset + cc*sin(sunset) ) #Hemmings code
# These agree except for factor out front (see personal comm. for unit change)

   I0<- 0.375*(1 - 0.7*C)*Isn


    if ((I0*alpha*VP) <= 0) {
      flowNP<-0

   } else {
     bet<- (VP*tau)/(I0*alpha)
```

```
    atten<- (kW + P*kP)*M

    attenmod<-atten

    bet2<- bet*exp( atten )
    outfront<- (360*tau*2*VP/atten)
    Ibar<- phint2(bet) -  phint2(bet2)

    Jbar<- outfront*Ibar
    Q<- N/(kN+N)

    flowNP<- P*Jbar*Q
}

 Pdeep<-phiP*P
 pzdenom<-kG+P
 flowPZ<- g*Z*P/pzdenom

 Zdetr<- (1-beta)*flowPZ
 epstmp<- epsilon*Zdetr

 flowZN<- mu*Z + Zdetr - epstmp

 Zdeep<-phiZ*Z^2 + epstmp

 hstar<-max(dMdt, 0)
 Common<-(hstar+m)/M

 fluxP<- Common*P

 fluxZ<- Common*Z

 Test.Nref<-Nref*log(b*M+1) - N

 if ( Test.Nref<=0  ){
   fluxN<-0
 } else {
   fluxN<- Common*Test.Nref
 }

 STATE<- matrix( c( flowNP - flowPZ - Pdeep - fluxP,
```

```
                    flowPZ - flowZN - Zdeep - fluxZ,
                    flowZN - flowNP + fluxN ), ncol=1 )

  if (deriv.order>0){

  dPdTheta<-Ymat[1, 2:(1+pact)]
  dZdTheta<-Ymat[2, 2:(1+pact)]
  dNdTheta<-Ymat[3, 2:(1+pact)]

   if ((I0*alpha*VP) <= 0) {
##   flowNP<-0
     dflowNP<-NULLVEC

  } else {
##  bet<- (VP*tau)/(I0*alpha)
    dbet<- NULLVEC
    if (is.element(3,act)){
      dbet[RANK[3]]<- tau/(alpha*I0) }
    if (is.element(4,act)){
      dbet[RANK[4]]<- -VP*tau/(alpha^2*I0) }

##  atten<- (kW + P*kP)*M
    datten<- dPdTheta*M*kP

    if (is.element(5,act)){
      datten[RANK[5]]<-datten[RANK[5]] + M }

    if (is.element(6,act)){
      datten[RANK[6]]<-datten[RANK[6]] + P*M }

##  bet2<- bet*exp( atten )
    dbet2<- exp(attenmod)*( dbet + bet*datten  )

 #   outfront<- 2*VP/atten
    doutfront<- -VP*datten
    if (is.element(3,act)){ # Extra term for theta=VP
      doutfront[RANK[3]]<- doutfront[RANK[3]] + attenmod }
    doutfront<-doutfront/attenmod^2 #Division of whole vector last

    doutfront<- (360*tau*2)*doutfront  # Don't forget the multiplier!
```

```
## Ibar<- phint2(bet) -  phint2(bet2)
   dIbar<- dphint(bet)*dbet - dphint(bet2)*dbet2

## Jbar<- outfront*Ibar
   dJbar<- outfront*dIbar + doutfront*Ibar

## Q<- N/(kN+N)
   dQ<- dNdTheta*kN
   if (is.element(2,act)){
     dQ[RANK[2]]<- dQ[RANK[2]] - N } # Extra term for theta=kN
   dQ<-dQ/(N+kN)^2 #Do division of everything last

## flowNP<- P*Jbar*Q
   dflowNP<- dPdTheta*Jbar*Q + P*( Q*dJbar + dQ*Jbar)

   }


## Pdeep<-phiP*P
   dPdeep<-phiP*dPdTheta
   if (is.element(1,act)){
     dPdeep[RANK[1]]<-dPdeep[RANK[1]] + P } # Extra term for theta=phiP

## pzdenom<-kG+P
## flowPZ<- g*Z*P/pzdenom
   dflowPZ<- g * ( dZdTheta*P*pzdenom + Z*dPdTheta*kG )

   if (is.element(10,act)){ # Extra term for theta=g
     dflowPZ[RANK[10]]<- dflowPZ[RANK[10]] + (Z*P*pzdenom) }

   if (is.element(11,act)){ # Extra term for theta=kG
     dflowPZ[RANK[11]]<- dflowPZ[RANK[11]] - (P*Z*g) }

   dflowPZ<-dflowPZ/pzdenom^2
   # Division is very last operation to combat numerical errors

## Zdetr<- (1-beta)*flowPZ
   dZdetr<- (1-beta)*dflowPZ
   if (is.element(7,act)){ # Extra term for theta=beta
     dZdetr[RANK[7]]<- dZdetr[RANK[7]] - flowPZ }
```

```
## epstmp<- epsilon*Zdetr
   depstmp<- epsilon*dZdetr
   if (is.element(12,act)){ # Extra term for theta=epsilon
     depstmp[RANK[12]]<- depstmp[RANK[12]] + Zdetr }

## flowZN<- mu*Z + Zdetr - epstmp
   dflowZN<- mu*dZdTheta + dZdetr - depstmp

   if (is.element(8,act)){ # Extra term for theta=mu
     dflowZN[RANK[8]]<- dflowZN[RANK[8]] + Z }

## Zdeep<-phiZ*Z^2 + epstmp
   dZdeep<-phiZ*2*Z*dZdTheta + depstmp

   if (is.element(9,act)){ # Extra term for theta=phiZ (not theta=phiP!)
     dZdeep[RANK[9]]<-dZdeep[RANK[9]] + Z^2 }

## hstar<-max(dMdt, 0)
## Common<-(hstar+m)/M

## fluxP<- Common*P
   dfluxP<- Common*dPdTheta
   if (is.element(15,act)){ #Extra term for theta=m
     dfluxP[RANK[15]]<- dfluxP[RANK[15]] + P/M }

## fluxZ<- Common*Z
   dfluxZ<- Common*dZdTheta
   if (is.element(15,act)){ #Extra term for theta=m
     dfluxZ[RANK[15]]<- dfluxZ[RANK[15]] + Z/M }

## Test.Nref<-Nref*log(b*M+1) - N

    if ( Test.Nref<=0  ){
     dfluxN<-NULLVEC
   } else {
##   fluxN<- Common*Test.Nref
     dfluxN<- -Common*dNdTheta
     if (is.element(13,act)){ # Extra term for theta=Nref
       dfluxN[RANK[13]]<- dfluxN[RANK[13]] + Common*log(b*M+1) }

     if (is.element(14,act)){ # Extra term for theta=b
```

```
      dfluxN[RANK[14]]<-dfluxN[RANK[14]]+(hstar+m)*Nref/(b*M+1) }

    if (is.element(15,act)){ # Extra term for theta=m
      dfluxN[RANK[15]]<- dfluxN[RANK[15]] + Test.Nref/M }
  }

  dY<-rbind(  dflowNP - dflowPZ - dPdeep - dfluxP,
              dflowPZ - dflowZN - dZdeep - dfluxZ,
              dflowZN - dflowNP + dfluxN   )

  return( cbind( STATE, dY ) )

} else { return(STATE) }

}


RKCK<-function( ymat, dord ){ #deriv.order

  t<-ymat[[1]]; Y<-ymat[[2]];

  K<-vector("list",6)

  K[[1]]<-h*fun(t, Y, dord )
  K[[2]]<-h*fun( t+Ca[2]*h, Y + Cb[2,1]*K[[1]], dord )
  K[[3]]<-h*fun( t+Ca[3]*h, Y + Cb[3,1]*K[[1]] + Cb[3,2]*K[[2]], dord )
  K[[4]]<-h*fun( t+Ca[4]*h, Y + Cb[4,1]*K[[1]] + Cb[4,2]*K[[2]] +
          Cb[4,3]*K[[3]], dord )
  K[[5]]<-h*fun( t+Ca[5]*h, Y + Cb[5,1]*K[[1]] + Cb[5,2]*K[[2]] +
          Cb[5,3]*K[[3]] + Cb[5,4]*K[[4]], dord )
  K[[6]]<-h*fun( t+Ca[6]*h, Y + Cb[6,1]*K[[1]] + Cb[6,2]*K[[2]] +
          Cb[6,3]*K[[3]] + Cb[6,4]*K[[4]] + Cb[6,5]*K[[5]], dord  )

  hdYdx<- Cc[1]*K[[1]]+ Cc[2]*K[[2]] + Cc[3]*K[[3]] + Cc[4]*K[[4]] +
          Cc[5]*K[[5]] + Cc[6]*K[[6]]

  Y<- Y + hdYdx

  DeltaErr<-dCc[1]*K[[1]] + dCc[2]*K[[2]] + dCc[3]*K[[3]] + dCc[4]*K[[4]] +
            dCc[5]*K[[5]] + dCc[6]*K[[6]]
```

```
  list( tnew=t+h, Y=Y, DeltaErr=DeltaErr, hdYdx=hdYdx)

}



RKAS<-function( y, dord=1 ){


Try<-RKCK(y, dord);

#yscal<-Try[[2]]; DeltaErr<-Try[[3]]; hdYdx<-Try[[4]];
yscal<-Try$Y; DeltaErr<-Try$DeltaErr; hdYdx<-Try$hdYdx

if (adapt=="all"){
  err<- abs( DeltaErr ) / (  abs(yscal)  +  abs(hdYdx)  +  TINY  )
} else { # Only check error for state variables not derivatives
  err<- abs( DeltaErr[,1] ) / (  abs(yscal[,1])  +  abs(hdYdx[,1])  +  TINY  )
}

err<-err/EPS

errmax<-max(err, na.rm=T)

if ( any( is.na(err) ) ) {
cat( paste("err has NA, errmax=", errmax, "\n") );
errmax<-10*max(errmax,1) }

w<-which( !abs(err)==Inf )
if ( !all(w) ) { errmax<-10*max(err[w],1) }

if (errmax > 1){
# Truncation error too large, reduce stepsize.

  nbad<<-nbad+1

  htmp<-max(SAFETY*h*errmax^PSHRINK, 0.1*h) #No more than a factor of 10

    if (tim==tim+htmp) {
        UNDERFLOW<<-TRUE
        print("Stepsize Underflow")
```

```
        print(y)
        return ( list( Try[[1]], Try[[2]] ) )
    } else {
      h<<-htmp;
      return ( RKAS(y, dord) )
    }

} else {

   ngood<<-ngood+1
   h<<-min(SAFETY*h*errmax^PGROW, 5*h, HMAX);
   # Maximum factor of 5 increase AND Don't let h go above HMAX
   tim<<-Try[[1]]
   return ( list( tim, yscal  ) )

}

}



# Initial conditions
  IC<-matrix(c(0.02, 0.002, 1), ncol=1)

  if (deriv.order>0){ # Add init condits for derivs
    IC<-cbind( IC, matrix(0, nrow=3, ncol=pact) )
  }

  noutput<-nrow(IC);
  IC<-list(0, IC ); # First component of IC is t=0

nruns<-nrow(inp)


for (r in 1:nruns) {

x<-inp[r,]

for (i in 1:15){  assign(pars[i], unlist(x[i]), envir=.GlobalEnv)  }
```

```
h<-HINITIAL


len<- ncol(IC[2]) # (p+1)
Ymatrix<-c(  1, IC[[1]], as.vector( t(IC[[2]]) )  )

 # 3*16 + 2
 # First elt is period, second is time.

per<-0
nsteps<-0
nbad<<-ngood<-0
UNDERFLOW<<-FALSE

while ( all( per<nperiod, nsteps<=MAXnsteps ) ) {

  per<-per+1;
  Mcur<<-Ccur<<-1; tim<-0; #nsteps<-0; Switched back to 1 for R
  h<-HINITIAL; # Add it in here for control

  if (per==1) {
    y<-IC;
  } else {
    y[[1]]<-0;
  }  # Reset time here

  nit<-0;
  while ( all(y[[1]]<365, nsteps<MAXnsteps, !UNDERFLOW) ) {
       nit<-nit+1
       nsteps<-nsteps+1
       if ( (y[[1]]+h) > 365 ){ h<<- 365-y[[1]] }
          # Last step takes you exactly to 365
       y<-RKAS(y, deriv.order)
       Transy<-c(  per, y[[1]], as.vector( t(y[[2]]) )  )
       Ymatrix<-rbind(Ymatrix, Transy)
  }

  if (nsteps>=MAXnsteps) {
    cat( paste("nsteps > MAXnsteps, t =", y[[1]],"\n") )
  }
```

```
}


if ( all(nsteps <= MAXnsteps, !UNDERFLOW) ){
  Ymatrix<-Ymatrix[ which(Ymatrix[,1]==2), ]
  colnames(Ymatrix)<-NAM
  rownames(Ymatrix)<-NULL
  fnam<- paste(runnames, r, sep="." )
  fp<-file.path("~dma0psc","research","ocean","simulators","DerivativeRuns")
  write.table( Ymatrix, file=file.path(fp, fnam), quote=FALSE, row.names=FALSE)
}

ctime<- strsplit(date(), split="")[[1]][ c(12,13,15,16,18, 19) ]
print(paste("Run", r, ", ng=", ngood, ", nb=", nbad, sep=""))
NSTEPS<-rbind(NSTEPS, c(r, ngood, nbad, as.integer(ctime)) )
# END BIG FINAL LOOP
}


return ( NSTEPS )

} # End Solve.PZN
```

Appendix C

Emulator updating code

```
is.pd<-function(M){
  all(eigen(M)$values>=0)
}

g<-function(z, P){

  n<-nrow(z)
  q<-nrow(P)
  gx<-array(1, dim=c(q, n))

  for (i in 1:q){

    if ( all(is.na(P[i,])) ) { gx[i,]<-0
    # Replaces initial 1's in matrix by 0's

    } else {

      w<-which(!P[i,]==0) # Doesn't count NA values
        if (any(w)){
          for (j in w){ gx[i,]<-gx[i,]*z[,j]^P[i,j] }
        }
    }

  }
  gx
}
```

```
s.prior<-function(Priors, x){

##Computes Prior Specs for objects involving S

#Priors
EB<-Priors$EB; VarB<-Priors$VarB; gp<-Priors$gp;
sigmaEp<-Priors$sigmaEp; vdelta<-Priors$vdelta; sigmaDel<-Priors$sigmaDel

  if (any( c(is.null(EB), is.null(VarB)))){
    Es<-VgBg<-0
  } else {

  gx<-g(x, gp)
  Es<-EB%*%gx

  gVarBg<-tensor(VarB,gx,3,1) # k,q,L
  gVarBg<-aperm( gVarBg, c(2,1,3))

  other1<-function(j){
   gx[,j]%*%gVarBg[,,j]
  }

# Can tensor it and then apply(,,diag) over the k,k dimension, but
# this means you first have to allocate a large vector (too large for R!)

 VgBg<-sapply( 1:nrow(x), other1 )

 if (length(dim(VgBg))){ VgBg<-t(VgBg) } #get dim NULL if 1 output


#  VgBg<-function(gx, VarB) tensor( gx, tensor(VarB, gx, 3,1), 1,2)

#  VgBg<-apply(gx, 2, function(z) VgBg(z,VarB))
#  VgBg<-t(VgBg)

##  VgBg<-tensor( g, tensor(VarB, g, 4,1), 1,2)
##  gives dim 1,k,k,l
##  VgBg<-aperm(VgBg, c(2,1,3,4))
#  VgBg<-tensor( gx, tensor(VarB, gx, 3,1), 1,2)
#  VgBg<-aperm(VgBg, c(2,1,3))
```

```
#  VgBg<-apply(VgBg, 1, diag) # Read off the x variances
##   l<-ceiling( sqrt(nrow(x)) )
##   VE<-VarEp(x)
##   VD<-outer( sigmaDel, diag(l) )
##   VD<-aperm(VD, c(1,3,2))
  }

  #VE<-outer( rep(1, nrow(x)), sigmaEp ) #works for multiV but slower for uniV
  #VD<-outer( rep(1, nrow(x)), sigmaDel )
  VE<-matrix(sigmaEp, ncol=length(sigmaEp), nrow=nrow(x), byrow=T )
  VD<-matrix(sigmaDel, ncol=length(sigmaDel), nrow=nrow(x), byrow=T )

  list( Esx = Es, Varsx = VgBg + VE + VD, VargBg = VgBg, VarEpx = VE )

}


AdjustBeliefs<-function (Priors, Data, x) {

#Priors
EB<-Priors$EB; VarB<-Priors$VarB; gp<-Priors$gp; actx<-Priors$actx;
theta<-Priors$theta; sigmaEp<-Priors$sigmaEp;
vdelta<-Priors$vdelta; sigmaDel<-Priors$sigmaDel

#Data
X<-Data$x; y<-Data$y; dydx<-Data$dydx
k<-ncol(y); n<-nrow(y)
p<-ncol(gp); q<-nrow(gp)

if (VER=="vN"){ #NULL
  #y<-cbind(y[,k],y[,k] )
   y<-y[,k,drop=F]
  if (length(dydx)){
    #dydx<-array(NA, dim=c(n,2,p))
    #dydx[,1,]<-dydx[,2,]<-Data$dydx[,k,,drop=F] }
     dydx<-Data$dydx[,k,,drop=F] }
    if (length(dydx)){
      #dydx<-ifelse(abs(dydx)>20, sign(dydx)*(17+log(abs(dydx))), dydx)
      #17+log(20)=20, 12.3+log(15) = 15
    }
  #k<-2
```

```
   k<-1
} else {#v2
  y<-y[,1:(k-1),drop=F]
  if (length(dydx)){
    dydx<-Data$dydx[,1:(k-1),,drop=F]
    # Try rescaling outliers ############################
    #dydx<-ifelse(abs(dydx)>5, sign(dydx)*(3.4+log(abs(dydx))), dydx)
    #dydx<-ifelse(abs(dydx)>7.5, sign(dydx)*(5.5+log(abs(dydx))), dydx)
    #log(10)+7.7=10, log(5) + 3.4 = 5, log(2.5)+1.58 = 2.5
   }
  k<-k-1
}




if (!all(p==length(actx), p==ncol(x))){
  stop("Problem with p dim")
}


if ( max(abs(X), na.rm=T) > 1 ) {
   stop("X should be transformed onto [-1, 1]")  }


if (length(dydx)){
  dydx<-aperm(dydx, c(2,1,3))
  dim(dydx)<-c(k, n*length(actx))
}


S<-cbind(t(y), dydx) #dim k,n(p+1)

Xact<-X[, actx, drop=FALSE]
# one actx for all output (called seperately for P and Nw)
X<-Xact

Xactexp<-t( sqrt(theta)*t(Xact) )

#Calculate dimension of simulator from priors
```

```
 if (any( c(is.null(EB), is.null(VarB)))){ #no global terms
    ES<-VarGB<-0
  } else {

dgp<-array( gp, dim=c(dim(gp), p) )

for (dx in 1:p){

    P<-dgp[,,dx]

      for (i in 1:q){

        if (P[i,dx]>0){ #if the power wrt x[dx] of the qth fn is > 0
          # Either subtract one off i,dx entry or make whole row NA
          P[i,dx]<-P[i,dx]-1
        } else {
          P[i,]<-NA }

      }

    dgp[,,dx]<-P

}

G<-g(Xact, gp) # G includes derivatives but g doesn't...
if (length(dydx)){
  for (i in 1:p){
    G<-cbind( G, g(Xact, dgp[,,i]) )  }
  }

# Gives dimG = q , n(p+1)


ES<-EB%*%G

CovBS<-tensor(VarB, G, 3, 1)
#gives dim k q n


VarGB<-tensor(G, CovBS, 1, 2)
#gives dim n,k,n
```

```
VarGB<-aperm(VarGB, c(2,1,3))
#gives dim k,n,n

check<-apply(VarGB, 1, is.pd )
#if (!all(check)) {
#  print("VarGB not pd")
#  return (VarGB)
#}

} # End global part


VarEp<-function(x){
  VarEp<-sigmaEp%o%exp( -as.matrix(dist(x))^2 )
  #VarEp<-sigmaEp%o%exp( -theta*as.matrix(dist(x))^2 )
  #produces dim k,1,1
  #VarEp<-aperm(VarEp, c(1,3,2))
  VarEp
}


VarD<-diag(1,n)

if (length(dydx)){
  VarD<-array( 0, dim=c(n*(p+1), n*(p+1)) )
  VarD[1:n, 1:n]<-diag(1,n)
}

VarD<-outer( vdelta, VarD )
#gives dim k,n,n
#VarD<-aperm(VarD, c(1,3,2))


mvdist<-function(y, z=y){
  y<-as.matrix(y)
  z<-as.matrix(z)
  n<-nrow(y)
  cc<-rbind(y,z)
  dd<-as.matrix(dist(cc))
  dd[1:n, -(1:n), drop=FALSE]
}
```

```
#VarE<-exp(-theta*as.matrix(dist(Xact))^2)
VarE<-exp(-as.matrix(dist(Xactexp))^2)

if (length(dydx)){
  lin<-apply( t( theta*t(Xact) ), 2, function(z) outer(z, z, "-") )
  #dim is n^2,p
  dim(lin)<-c(n, n, p)
  pcopiesVarE<-array(VarE, dim=c(dim(VarE),p)) #pcopiesVarE[,,1]-VarE TICK
  Cfdx<- 2*lin*pcopiesVarE # 2theta

   int<-array(NA, dim=c(n,p,n,p))
    for (i in 1:p){
      for (j in 1:p){
         int[,i,,j]<- lin[,,i]*lin[,,j]
         # 2*theta as have exp(-theta) instead of exp(-theta/2)
      }
    }

  p2copiesVarE<-array(VarE, dim=c(dim(VarE),p,p))
  p2copiesVarE<-aperm(p2copiesVarE, c(1,3,2,4))
  ZeroOne<-outer( diag(theta,p), array(1,dim=c(n,n)) )
  ZeroOne<-aperm(ZeroOne, c(4,2,3,1))
  Cdxdx<- 2*( ZeroOne - 2*int )*p2copiesVarE
  np<-n*p
  dim(Cdxdx)<-c(np, np)
  dim(Cfdx)<-c(n, np)

  firstrow<-cbind(VarE, Cfdx)

  VarE<-rbind( firstrow, cbind(t(Cfdx), Cdxdx) ) #dim n(p+1), n(p+1)
}

#Check
if (!is.pd(VarE)) stop("VarE not pd")

VarE<-sigmaEp%o%VarE
#VarE<-aperm(VarE, c(1,3,2))
#gives dim n,k,n #NB n->n(p+1) for derivs
VarS<-VarGB + VarE + VarD
```

```
# Outputs are uncorrelated
IVarS<-array(dim=dim(VarS))

for (i in 1:k){
  IVarS[i,,] <- chol2inv( chol(VarS[i,,]) )
}



diff<-S-ES

R<-array( dim = c( k, dim(VarE)[3] ) )
ESB<-array(dim=dim(EB))
VarSB<-array(dim=dim(VarB))
sigmaEp.adj<-c()

#Update beliefs about B
for (i in 1:k){
  CovBStmp<-CovBS[i,,]
  IVarStmp<-IVarS[i,,]
  R[i,]<-IVarStmp%*%diff[i,]

  if ( is.null(dim(CovBStmp)) ){
    dim(CovBStmp)<-c(length(CovBStmp), 1)
  }
  if ( is.null(dim(IVarStmp)) ) {
    dim(IVarStmp)<-c(1,1)
  }

  ESB[i,]<-CovBStmp%*%R[i,]
  VarSB[i,,]<- CovBStmp%*%IVarStmp%*%t(CovBStmp)
}

ESB<-EB+ESB
VarSB<-VarB-VarSB

# Efficient code if only want post Var for each x, not post Cov matrix.
# NB outer still quick for 1e5 grid points and 100 data points
# So e.g. 17^4 = 83,521

L<-nrow(x)
```

```
lin<-array(NA, dim=c(L,n,p))
quad<-array(0, dim=c(L,n))
for (i in 1:ncol(Xact)){
  tmp<-outer( x[,i], X[,i], "-")
  lin[,,i]<-theta[i]*tmp
# quad[,,i]<-sqrt(theta[i])*tmp
  quad<-quad + theta[i]*tmp^2
}

CEpxS<-exp( -quad )

#CEpxS<-exp( -mvdist(xexp,Xactexp)^2 )
# Do this if want posterior cov between points but very expensive.

if (length(dydx)){
  pcopies<-array( rep(CEpxS, p), dim=c(dim(CEpxS),p) )
  pcopies<- 2*lin*pcopies
  dim(pcopies)<-c(L, n*p)
  CEpxS<-cbind(CEpxS, pcopies)
  rm(pcopies) #get rid of large object
}

#Do we need to do this outer product HERE in univariate case?
CEpxS<-sigmaEp%o%CEpxS

# dim k,L,np whereas R array is k,np
# IVarS is k,np,np

ESEpx<-array(dim=c(k,L))
RVarSEpx<-array(dim=c(k,L))
CSBEpx<-array(dim=c(k,q,L))

# Loop over outputs quicker than k*k output cov matrix with off diags set to 0
for (i in 1:k){
  ESEpx[i,]<- CEpxS[i,,]%*%t(R[i,,drop=F])
  #dim gets dropped for n=1, no deriv
  if (L==1) {
    Rtmp<-IVarS[i,,]%*%as.matrix(CEpxS[i,,])
  } else {
    Rtmp<- IVarS[i,,]%*%t(CEpxS[i,,])
  }#Rtmp is kn, L
```

```
  #dim gets drop for L=1
  CSBEpx[i,,]<- -CovBS[i,,]%*%Rtmp

  RVarSEpx[i,]<-sapply(1:nrow(x), function(j) CEpxS[i,j,]%*%Rtmp[,j]   )
# Much quicker to zip down than to use diag
}

VarEpx<-matrix(sigmaEp, nrow=k, ncol=L)
VarSEpx<-VarEpx - RVarSEpx

gx<-g(x, gp) #q,L
gVarSBg<-tensor(VarSB,gx,3,1) # k,q,L
gVarSBg<-aperm(gVarSBg, c(2,1,3))

other1<-function(j){
  gx[,j]%*%gVarSBg[,,j]
}

other2<-function(j){
   B1<-tensor( CSBEpx[,,j,drop=F], gx[,j], 2, 1 )
   B2<-tensor( aperm(CSBEpx[,,j,drop=F],c(2,1,3)), gx[,j], 1, 1 )
   B1+B2
}

gVarSBg<-sapply( 1:nrow(x), other1 )
CovSgBEpx<-sapply( 1:nrow(x), other2 )

mx<-ESB%*%gx + ESEpx

vd<-matrix(vdelta, nrow=length(vdelta), ncol= nrow(x) )
sx<-gVarSBg + CovSgBEpx + VarSEpx + vd

  #Check
  w<-which(sx<0)
  if (length(w)){
   wx<-apply(sx, 1, function(z) which(z<0) )
   wx<-unique(unlist(wx))
   cat( paste("No. x pts with less than 0 is:",
           length(wx), "min val:", min(sx),''\n'' ) )
  }
```

```
  sigmaEp.adj<-c()
  diff.adj<-S -ESB%*%G
  sigmaEp.adj<-c()
  for (i in 1:k){
    sigmaEp.adj<-c(sigmaEp.adj, diff[i,]%*%IVarS[i,,]%*%diff[i,] )
  }
  sigmaEp.adj<-sigmaEp.adj/n

  list( Priors=Priors, ESB=ESB, VarSB=VarSB, mx=mx, sx=sx, gVarSBg=gVarSBg,
      VarSEpx=VarSEpx, Xact=Xact, sigmaEp.adj=sigmaEp.adj )

}
```

Appendix D

Implausibility code

```
Imp<-function(Eval, Z, Sigmaeta, Escal=(0:5)^2){

 DIMS<-dim(Eval$mx)
 nr<-DIMS[1]; nc<-DIMS[2]

 d<-Z[, "y"]
 SigmaeP<-Z[, "Sigmae"]
 n<-length(Escal)

 # duplicate for each observation
 onevec<-rep(1,nc)
 dx<-outer( d, onevec)
 SigmaePx<-outer( SigmaeP, onevec )
 Sigmaetax<-outer( Sigmaeta, onevec )

 # duplicate for different discrepany scalings
 Edif <- abs( Eval$mx - dx )
 Edif <- outer(Edif, rep(1,n))
 Sigmadif <- outer(Eval$sx+SigmaePx, rep(1,n)) + outer(Sigmaetax, Escal )
 #Trace<-apply( Sigmadif, 2, sum)
 Sigmadif<- sqrt( Sigmadif )
 ImpByOut<- Edif/Sigmadif

 #MVEdif<-sapply(1:nc, function(z) t(Edif[,z])%*%Edif[,z] )
 #MVImp<-MVEdif/Trace

 #wmax<-apply( ImpByOut, 2, which.max ) #NB gives FIRST max
 #over index 2 here as sigmaeta fixed
 wmax<-apply( ImpByOut, c(2,3), which.max )
```

```
# hist(wmax)
#tells us which of the outputs is most useful for calibration using Impmax

# Emax<-sapply( 1:nc, function(i) Edif[wmax[i],i] )
# Smax<-sapply( 1:nc, function(i) Sigmadif[wmax[i],i] )
# Take these two out to save on storage
# Imax<-sapply( 1:nc, function(i) ImpByOut[wmax[i],i] )

Imax<-apply( ImpByOut, c(2,3), max)

# list( ImpByOut=ImpByOut, wmax=wmax, Edifmax=Emax, Sigmadifmax=Smax,
#       Impmax=Imax, MVEdif=MVEdif, Trace=Trace )

list( ImpByOut=ImpByOut, wmax=wmax, Impmax=Imax )


}

Imp2D<-function(Implaus, xpair, out=1:13){

  x<-Implaus$x
  Impmax<-Implaus$Impmax
  p<-ncol(x)
  l<-nrow(x)

  if ( length(xpair) >= p )
  stop("Need at least one var to minimise over")

  x<-as.matrix(x)
  minover<-setdiff(1:p, xpair)


  # Minimise over other inputs

  xp2D<-unique(x[,xpair,drop=F])

 X2D<-as.matrix(x[, xpair,drop=F])
 # res<-as.integer( sqrt(nrow(xp2D)) )

min2<-function(z){
```

```
  w<-1:nrow(x)
  for (i in 1:length(xpair)){
    wi<-which( x[,xpair[i]]==z[i] )
    w<-intersect(w, wi )
  }
  W<-which.min(Impmax[w])
  MEAN<-mean(Impmax[w])
  c( w[W], MEAN )
}

Impmax2D<-apply( xp2D, 1, min2 )

Impmax<-Impmax[Impmax2D[1,]] #just want column 1 of w[W]
Impmean<-Impmax2D[2,]

# Can send out variance surface here easily too
list(  Impmax=Impmax, x=xp2D, xpair=xpair, Impmean=Impmean  )


}
```

Appendix E

Miscellaneous code

```
MSDM<-function(Theta0, STAS, nrun){

for (STA in STAS){

runnames<-paste("HYPER", nrun, "s", STA, sep="")
load(file=paste(runnames, ".Rdata", sep=""))

IN<-1:nrun
X<-S$x
X<-Transform(X, R=1)
X<-X[IN,]
X<-as.data.frame(X)
Y<-S$y
dY<-S$dydx
k<-dim(S$y)[2]

new.theta<-c()
Rsq<-c()
f.fits<-list()
  for (out in 1:k){

    if (out==k){
      act<-c(1,4,6,13)
      f<-lm( Y[IN,out,2] ~ X1+X4+X6+X13, data=X)
    } else {
      act<-c(1,4,6,10 )
      f<-lm( fmla( "Y[IN,out,2]", act, int2nd=T), data=X)
    }
```

```
    f.fits[[out]]<-f
    Rsq<-c(Rsq, summary(f)$r.squared)


    mult[c(5,14)]<-0
    dYtmp<-matrix(mult, nrow=dim(dY)[1], ncol=15, byrow=T)*dY[,out,]

    efit<- fit(f, act, theta=Theta0 )

    esttheta<-function(inp, VEPS){
      y<- paste( "dYtmp[," , inp, "]", sep="")
      if (out==k){
        dfdx<-lm( dYtmp[, inp] ~ 1, data=X)
      } else {
         dfdx<-lm( fmla( y, setdiff(act,inp)), data=X)
      }
      DOF <- dfdx$df.residual #degrees of freedom
      e<-residuals(dfdx)
      s2 <- drop((t(e)%*%e)/DOF) #Estimate of 2*theta_{i}sigma_{e}^2
      #New theta estimate
      s2/(2*max(VEPS,0.01)) #correct if VEPS too small

    }

    thetaobj<-sapply( act,function(z) esttheta(z,efit$veps))
    new.theta<-rbind(new.theta, thetaobj)
)

}

lastP<-k-1
MEAN<-apply(new.theta[1:lastP,], 2, mean)
MED<-apply(new.theta[1:lastP,], 2, median)
MAX<-apply(new.theta[1:lastP,], 2, max)
MAX.NOEND<-apply(new.theta[2:(lastP-1),], 2, max) #ignore first and last P
new.theta<-rbind( new.theta, MED, MEAN, MAX, MAX.NOEND )
write.table(new.theta, file=paste("new.theta", STA, sep=""))
colnames(new.theta)<-paste("& \theta_{", 1:4, "}", sep="")
rownames(new.theta)[1:k]<-c(paste("P_{", 1:(k-1), "} &"), "N_{w} &")
print( round(new.theta,2) )
```

```
print(Rsq)

for (out in 1:k){
  if (out==k){ NEW.THETA<-new.theta[k,] } else { NEW.THETA<-MAX.NOEND }
  compare<-rbind( unlist(fit(f.fits[[out]], act, theta=Theta0 )),
                  unlist(fit(f.fits[[out]], act, theta=NEW.THETA )) )
  print( round(compare,3) )
}

}

} #End MSDM


Best.hypercube<-function( n, w,  ntrys=1e3, tovary=setdiff(1:15, c(5,14))){

d<-length(tovary)
cur<-rep(1, length(w)); themax<-rep(0,length(w))
#w=0 corresponds to mimimising C13, w=1 to minimising C5
CORRmax<-CORR<-matrix(NA, ncol=13, nrow=length(w))
CORRsum<-matrix(NA, ncol=2, nrow=length(w))# ncol=2: sum to 5 and sum to 13

for (i in 1:ntrys){

  HYPER<-hypercube(n, d, range=c(-1,1))
  corr<-cor( HYPER)
  diag(corr)<-0 #Don't want 1s to count is maximum
  corr<-apply(abs(corr), 1, max )
  corr<-sort(corr)
  tomin<-w*corr[5] + (1-w)*corr[13]

  wmin<-which(tomin<cur)
  wmax<-which(tomin>themax)

  if (length(wmin)){
    CORR[wmin,]<-corr; CORRsum[wmin,]<-c(sum(corr[1:5]),sum(corr))
    if (length(w)==1){ CUR<-HYPER } else { CUR<-NULL }
      # CUR<-HYPER makes sense only for w scalar
  } # end if (length(wmin))

  if (length(wmax)){
```

```
   CORRmax[wmax,]<-corr }

  cur<-ifelse(tomin<cur, tomin, cur)
  themax<-ifelse(tomin>themax, tomin, themax)

}

if (length(w)==1){
  CUR<-InvTrans(CUR, tovary, R=1)
  X<-matrix(XPrior$def, nrow=n, ncol=15, byrow=T)
  X[,tovary]<-CUR
} else { X<-NULL }

list(corr=CORR, X=X, corrmax=CORRmax, corrsum=CORRsum)
#crit is the biggest value - best if not x1,x4,x6,x10,x13 for tomin=13

}



fmla <- function(y, active, quad=F, int=F,, int2nd=F, extra=NULL){

  xnam<-paste("X", active, sep="")
  lin<-paste(xnam, collapse= "+")
  terms<-lin


  if (quad) {
     quad<-paste("I(", xnam, "^2)", sep="")
     quad<-paste(quad, collapse="+")
     terms<-paste(terms, quad, sep="+")
  }



 if ( all( c(int2nd, length(active)>1, !int) ) ){
      # NB before int gets changed into a string so !int makes sense
 matt<-c()
 for (i in 1:(length(active)-1)){
    newbit<-cbind(active[i], active[(i+1):length(active)])
    matt<-rbind(matt, newbit)
```

```
 }

  i2<-apply(matt, 1, function(z)  paste(paste("X",z, sep=""),collapse=":" )  )
  i2<-paste(i2, collapse="+")
  terms<-paste(terms, i2, sep="+")
}



  if ( all( c(int, length(active)>1) ) ) {
    int<-paste(xnam, collapse="*")
    terms<-paste(terms, int, sep="+")
  }


 if (length(extra)){
    terms<-paste(terms, extra, sep="+")
 }

  as.formula(paste( paste(y, " ~ ", sep=""), terms ))

}


Transform<-function(Z, l=XPrior$low, u=XPrior$upp, R=rad){

  # Simple Linear transform of Z onto range [-R, R] (Usually R=1)
  X<-array(dim=dim(Z), dimnames=list(NULL, colnames(Z)))
    for (j in 1:ncol(Z)){
      X[,j]<-( R/(l[j]-u[j]) )*( -2*Z[,j] + l[j]+u[j] )
    }

  X

  }


InvTrans<-function(Z, inp=1:15, l=XPrior$low, u=XPrior$upp,
                   d=XPrior$def, R=rad){

   # Gives X values back in original scale
```

```
   #eg to find refocused region in original scale

    X<-matrix( nrow=nrow(Z), ncol=ncol(Z) )

     for (j in seq(along=inp)){
       i<-inp[j]
       X[,j]<- 0.5*( l[i]+u[i] + Z[,j]*(u[i]-l[i])/R )
     }

    X

  }



Matern1D<-function(x, theta, nu){

  #rescale<-2*sqrt(nu)*abs(x)/theta # Santner theta scaling
  rescale<-2*sqrt(theta*nu)*abs(x) # My scaling (to coincide with Gauss)
  R<-rescale^nu * besselK(rescale, nu=nu)/( gamma(nu)*2^(nu-1) )
  ifelse(is.na(R), 1, R) # Get NA at x=0

}



Sim<-function( theta, n=4, X=c(), nu=NULL, deriv=F, LEN=101, PLOT=T,
               YLIM=NULL, XA=T, YA=T){

  require(MASS) # For multivariate normal sampling

  # Derivatives option only if data (length(X)>0) and Gaussian (nu=NULL)
  if ( any(!length(X), !is.null(nu)) ){ deriv<-F }

  x <- seq(-1, 1, len=LEN)

  xdist<-outer(x, x, "-") # For x a vector (1D)

  if (length(nu)){ # is.nul(nu) specifies Gauss
```

```
  Sigma <- Matern1D(xdist, theta, nu)
} else {
  Sigma <- exp(-theta * xdist^2 )
}

# Unconstrained by obs
if (!length(X)){
  y <- mvrnorm(n, rep(0, length(x)), Sigma)
} else {

  XX<-outer(X, X, "-")
  xX<-outer(x, X, "-")
  if (length(nu)){
    VX<-Matern1D(XX, theta, nu)
    CxX<-Matern1D(xX, theta, nu)
  } else {
    VX<-exp( -theta*XX^2 )
    CxX<-exp( -theta*xX^2 )

    if (deriv){
      VXdash<-2*theta*(1-2*theta*XX^2)*VX
      CXXdash<- 2*theta*XX*VX
      CxXdash<- 2*theta*xX*CxX
      VX<-rbind( cbind(VX, CXXdash), cbind(t(CXXdash),VXdash) )
      CxX<-cbind(CxX, CxXdash)
    }

  }

  D<-mvrnorm(1, rep(0, (deriv+1)*length(X)), VX)
  D<-as.matrix(D)
  IVX<-chol2inv(chol(VX))
  R<-CxX%*%IVX
  VarDS<-Sigma - R%*%t(CxX)
  EDS<-R%*%D
  y<-mvrnorm(n, drop(EDS), VarDS)

}

if (PLOT){
```

```
  if (is.null(YLIM)){ YLIM=range(t(y)) }
  matplot(x, t(y), type="l", ylim=YLIM, axes=all(XA,YA), lty=1, cex.axis=0.8)
  if (!all(XA,YA)){
    axis(side=1, at=c(-1,-0.5,0,0.5,1), lab=XA )
    axis(side=2, at=-3:3, lab=YA )
    box()
  }

  if (length(X)){ points(X, D[1:length(X)]) }

  } # endif PLOT

  if (length(X)){

    # BL estimates
    sig.est<-sum(D[1:length(X)]^2)
    if (deriv){
      theta.est<- sum(D[(length(X)+1):(2*length(X))]^2) / (2*sig.est)
    } else { theta.est<-NULL }

    return( list( X=X, sigmaEp=1, VX=VX, IVX=IVX, D=D, deriv=deriv,
                  sig.est=sig.est/length(X), theta.est=theta.est ) )
   }

}
```

Appendix F

Bessel functions

Bessel functions arise as the solution of a class of ordinary differential equations. In general, $K_\nu(x)$ in defined by an infinite power series. In the case that ν is equal to a half integer (that is, $\nu = n + 1/2$ for $n \in 0, 1, \ldots$), then

$$K_{n+1/2}(x) = e^{-x}\sqrt{\frac{\pi}{2x}}\sum_{k=0}^{n}\frac{(n+k)!}{k!(n-k)!}\frac{1}{(2x)^k} \tag{F.0.1}$$

Substituting this into the Matérn form

$$R(h) = \frac{1}{\Gamma(\nu)2^{\nu-1}}(2\sqrt{\nu\theta}|h|)^\nu K_\nu(2\sqrt{\nu\theta}|h|) \tag{F.0.2}$$

we get

$$R(h) = e^{-2\sqrt{\nu\theta}|h|}\left(b_0(|h|\sqrt{\theta})^n + b_1(|h|\sqrt{\theta})^{n-1} + \cdots + b_n\right) \tag{F.0.3}$$

where the coefficients are given by

$$b_j = \sqrt{\pi}\frac{\nu^{(n-j)/2}}{4^j\Gamma(\nu)}\frac{(n+j)!}{j!(n-j)!}, \quad \nu = n + 1/2,\ j = 0, 1, \ldots \tag{F.0.4}$$

It can be shown that $R(h) \to e^{-\theta h^2}$ as $\nu \to \infty$.

Appendix G

High order derivatives of the Gaussian covariance kernel

Letting x be scalar and writing $c(u) = e^{-u^2}$, where $u = \sqrt{\theta}(x - x')$, it is easy to see that

$$d^{m+n}c(x - x')/dx^m dx'^n = (-1)^n \sqrt{\theta}^{m+n} d^{m+n}c(u)/du^{m+n} \tag{G.0.1}$$

An expression for $d^k c(u)/du^k$, $k = m+n$, is then obtained by a simple rearrangement of the following definition for the Hermite polynomial of degree k (Gradstein and Ryshik, 1981),

$$H_k(u) = (-1)^n e^{u^2} d^k\left(e^{-u^2}\right)/du^k \tag{G.0.2}$$

Generalising for vector $x = (x_1, \ldots, x_p)$, we exploit the product form of the Gaussian covariance kernel which allows us to compute derivatives simply as the product of the corresponding scalar component derivatives. In particular, writing $u_i = \sqrt{\theta_i}(x_i - x'_i)$, $k = (k_1, \ldots, k_p)$ and defining the operator $\nabla_x^k = \frac{\partial^{k_1}}{\partial x_1} \cdots \frac{\partial^{k_p}}{\partial x_p}$, we obtain

$$\mathrm{Cov}[\nabla_x^k \epsilon(x),\ \nabla_x^{\bar{k}} \epsilon(x')] = \sigma_\epsilon^2 \exp^{-(x-x')^T \Theta (x-x')} \prod_{i=1}^{p} (-1)^{\bar{k}_i} \left(\sqrt{\theta_i}\right)^{k_i + \bar{k}_i} H_{k_i + \bar{k}_i}(u_i) \tag{G.0.3}$$

The advantage of writing (G.0.3) in this form is that it can be easily implemented using the recurrence relations for Hermite polynomials

$$H_0(u) = 1, \quad H_1(u) = 2u, \quad H_{k+1}(u) = 2uH_k(u) - 2kH_{k-1}(u) \tag{G.0.4}$$