Intermediate Variable Emulation: using internal processes in simulators to build more informative emulators*

Rachel Oughton[1], Michael Goldstein[1], and John Hemmings[2]

[1]Department of Mathematical Sciences, Durham University, South Road, Durham DH1 3LE, UK.
[2]Met Office, Fitzroy Road, Exeter, EX1 3PB

Abstract.

Complex systems are often modelled by intricate and intensive computer simulators. This makes their behaviour difficult to study, and so a statistical representation of the simulator is often used, known as an emulator, to enable users to explore the space more thoroughly. These have the disadvantage that they do not allow one to learn about the simulator's behaviour beyond its role as a function from input to output variables. We take a new approach, by involving the internal processes modelled within the simulator in our emulator.

We introduce a new technique, *intermediate variable emulation*, which enables a simulator to be understood in terms of the processes it models. This leads to advantages in simulator improvement and in calibration, as the simulator can be scrutinised in more detail and the physical processes can be used to refine the input space. The intermediate variable emulator also allows one to represent more complicated relationships within the simulator, as we show with a simple example.

We demonstrate the method using a simulator of the ocean carbon cycle. Using an intermediate variable emulator we are able to discover unrealistic behaviour in the simulator that would not be noticeable using a standard input to output emulator, and reduce the input space accordingly. We also learn about the sub-processes that drive the output, and about the input variables driving each sub-process.

1. Introduction.

Computer simulators are integral to many fields, including finance, environment and climate. Often, these simulators have many input variables, some of which represent properties of the system that are poorly understood or difficult to measure. This is particularly the case with marine biogeochemical models, where the values of the many input parameters are usually estimated in a field or lab experiment, if at all (22; 26; 33). When such simulators are computationally expensive and therefore slow to run, it becomes almost impossible to study their behaviour over their entire input space.

Bayesian emulation has helped tackle this problem, by building a statistical representation of a simulator using a set of 'training data' (simulator evaluations at some known points). An emulator represents a simulator as a regression surface (in terms of the simulator's input variables) and correlated error term, usually a Gaussian process. Training data and prior beliefs about the simulator's behaviour are combined so that the emulator can give its posterior distribution for the simulator's output at any point from the input space, from which we can make statements about our uncertainty. The emulator is almost always much faster than the simulator, making tasks such as prediction and calibration more feasible (12; 32; 24). For a discussion of the efficacy of Bayesian emulation in quantifying uncertainty, see Berger and Smith (3) As well as prediction, simulators can also be used to learn about the behaviour of the system (12; 33; 32) by using observations of output variables to restrict the input space to a plausible region, a process often known as *history matching*.

Simulators are often built from many carefully modelled sub-processes, representing various aspects of the system they model. The output is then governed by the activity of these sub-processes. An emulator treats a simulator as a function from input to output variables, and models the uncertainty about this function where it has not been evaluated. One exception is Drignei and Drignei (7), who use the underlying mathematical model in the simulator to inform their model of the output.

We present *Intermediate variable emulation*, a novel method that uses Bayesian emulation to learn more deeply about the simulator, by studying it at the sub-process level as well as using its input and output variables. Although here we have focused on using intermediate variable emulation to explore the behaviour of one simulator, it was originally developed with the goal of comparing multiple simulators of the same system.

*We are thankful to NERC for funding this research.

This can be seen in detail in Chapter 6 of (20).

In what follows, we introduce Intermediate Variable Emulation and demonstrate the methodology using an ocean carbon cycle simulator developed by Oschlies and Garçon (18). Using observed data and sensitivity analysis methods to understand a simulator better or to refine an input space is not new in ocean modelling, see for example (22; 25; 9), but we believe that our approach of looking at sub-processes within the simulator is an important contribution.

This paper is arranged as follows. An overview of emulation is given in section 2. Section 3 introduces the intermediate variable emulator and looks at some of the benefits of this method. In section 4 we apply intermediate variable emulation to an ocean carbon cycle simulator. Finally we summarise in section 5.

2. Emulation. An emulator is a statistical representation of our beliefs about a simulator $s(x)$ which, rather than giving single precise outputs $s(\mathbf{x})$ for a given set of input points $\mathbf{x}$, as the simulator does, gives a probability distribution for $s(\mathbf{x})$. Not only do we obtain the mean of the distribution, which is the emulator's approximation to $s(\mathbf{x})$, but in the variance of the distribution we also have a measure of confidence in the approximation.

If the emulator is faster to compute than the original simulator (this is almost always so) then we can obtain many more approximate values of the simulator than we could realistically obtain true values, and if the approximation is a good one then we can confidently use these values for our analysis.

A conventional structure for an emulator is to have a regression surface in x and a correlated error term modelled as a Gaussian process. The extent of the correlation is governed by some parameters commonly referred to as 'correlation lengths'. These govern how rough or smooth the function is. We use the posterior distribution

$$s(\tilde{\mathbf{x}}) \mid s(\mathbf{x}).$$

to emulate the simulator's behaviour at new inputs $\tilde{\mathbf{x}} = \{\tilde{x}_1, \ldots, \tilde{x}_m\}$ given the training data. See (19; 23) and (32) for more detail.

An emulator should meet two requirements (17):

1. At any of the points $x_1, \ldots, x_n$, where we already know exactly the output of the simulator, the emulator's distribution should match this value with zero variance, because we know with certainty that the simulator will always produce this same value. This property is guaranteed by the correlated error term. Note that this assumes a deterministic simulator, and that all the simulator's inputs appear in the emulator; in practice a 'nugget' term is often used, adding on some uncorrelated error.
2. For any input point x not in $\{x_1, \ldots, x_n\}$, the emulator's approximation should be plausible in view of our beliefs and the data we have, and the probability distribution should reflect our uncertainty about what the simulator may do at this point.

Verifying the first of these is simple. Checking the second requires more care, and some validation techniques for univariate emulators are described by Bastos and O'Hagan (1).

3. Intermediate Variable Emulation. The core premise of intermediate variable emulation is that a simulator can be thought of in terms of three stages, each represented by a set of variables: input variables, intermediate variables and output variables.

The input variables are the values input by the user to run the simulator. The output variables are those returned by the simulator, which an emulator aims to predict. These will usually be some summary of the raw simulator output, for example the mean or a multivariate summary of a time series. In standard emulation, as described in section 1, an emulator takes in the input variables and produces a probability distribution for the values of the output variables, treating the simulator as a black box. However, this results in a limited understanding of how the simulator models the system, and misses opportunities to make use of sub-processes.

Intermediate variables are a new construct, representing the states of the internal processes in a simulator. These are usually invested with some system meaning, for example the death or growth of a particular species, or a transfer or flux that takes place. Adopting these as intermediate variables is a natural way to be able to

better capture the behaviour of the simulator, and to better understand the relationship between inputs and outputs.

The set of intermediate variables can be listed for a simulator in such a way that if the values for all the intermediate variables are known, the input variables are no longer needed to calculate the simulator output. In this way they are similar to sufficient statistics: if two different input points produce identical values of the intermediate variables, they will also produce the same output value. It may be that some input variables must also be included as intermediate variables, if they cannot be separated out from the other intermediate variables. Instead of building an emulator of the output variables in terms of inputs, we emulate the intermediate variables in terms of the input variables, and then the output variables in terms of the intermediate variables.

It is worth noting that some of these analyses rely on a close working relationship between statistician and scientist. The intermediate variable emulation process may also be somewhat involved and awkward given the setup of many computer simulators. However, it is our hope that seeing the advantages of writing a simulator in such a way that these intermediate variables are easily accessible may encourage some modellers to do so. We believe there are several advantages to working with this structure, and we discuss them now.

3.1. Advantages of the intermediate variable structure.

Computation. Suppose we have a simulator $s(x)$, with one input variable x, and one intermediate variable u. We find that the best pair of emulators, subject to some constraints, is

$$s_u(u) = u^2 + \delta(u)$$
$$u(x) = x^2 + \gamma(x)$$

where $\delta(\cdot)$ and $\gamma(\cdot)$ are correlated error terms. An immediate advantage is that we have two separate error terms, which can be allowed to represent two different types of discrepancy. This is particularly advantageous where the correlated error term is accounting for a lot of behaviour. Concatenating these into a standard emulator form (ie. a function of x) gives

$$\begin{aligned} s_u(u) &= u^2 + \delta(u) \\ &= \left(x^2 + \gamma(x)\right)^2 + \delta\left(x^2 + \gamma(x)\right) \\ &= x^4 + 2x^2\gamma(x) + \gamma(x)^2 + \delta\left(x^2 + \gamma(x)\right). \end{aligned}$$

We have converted two clean, tractable emulators into a more complicated and intractable function. Thus the structure available to us using Intermediate Variable Emulation is not possible in the standard formulation.

One could also imagine that within the simulator the intermediate variable was transformed, for example

$$s(u) = \ln u + \delta(u)$$
$$u(x) = x^2 + \gamma(x).$$

Now the input-to-output version is

$$s(x) = \ln\left(x^2 + \gamma(x)\right) + \delta\left(x^2 + \gamma(x)\right),$$

which again is not a natural form for an emulator.

Insight. Emulating each sub-process separately allows us to see which input variables are driving which sub-process (captured by the intermediate variables), and in turn which sub-processes are driving which output variables. Suppose we find that input x_2 is not active for any intermediate variable. Then we can surmise that it will not be active for any of the output variables. Likewise, looking at which input variables are playing a strong role in the emulators of each intermediate variable may provide a helpful sanity check for the modeller: are these the inputs that should be important? Is the relationship between input and intermediate variable as they would expect? Are there any unexpected relationships?

History matching. Studying the behaviour of the intermediate variables also presents an opportunity to use system data relating to these sub-processes, if it is available. *History matching* is a technique where the emulator is used to rule out regions of input space corresponding to implausible regions of output space (31; 23; 35). The term originates from the oil industry, where model parameters are adjusted until they fit observed data, but the process was given a formal Bayesian approach by (4), and has since been adopted by the wider Bayesian computer models field. It is often the case that the emulator is more straightforward in this refined input region, since the global model tends to break down in difficult parts of the space (23).

If observations are available for the sub-processes modelled by the intermediate variables, these can be used to refine the input space. The two-stage nature of intermediate variable emulation also means that output observations could be used to history match to non-implausible regions of intermediate variable space, thus providing a further constraint on the input space.

Internal discrepancy modelling. Similarly to (29), who propose the notion of decomposing a computer model to better understand its subfunctions, we could add a model discrepancy term to each intermediate variable's emulator. For an input to output emulator, we would have

$$s^* = s(\mathbf{x}^*) + \delta_x^*,$$

where s^* is the real world value, $\mathbf{x}^*$ some 'best input value', and δ_x^* is the model discrepancy for the full simulator. For an intermediate variable emulator, we can instead specify sub-model discrepancies:

$$\begin{aligned} u_1^* &= \mathbf{u}_1(\mathbf{x}^*) + \delta_{u_1} \\ u_2^* &= \mathbf{u}_2(\mathbf{x}^*) + \delta_{u_2} \end{aligned}$$

where $\mathbf{u}_1(\cdot)$ and $\mathbf{u}_2(\cdot)$ are intermediate variables, and $\delta_{u_1}, \delta_{u_2}$ their associated discrepancies. This leads to

$$s^* = s_u\left(\mathbf{u}_1(\mathbf{x}^*) + \delta_{u_1},\ \mathbf{u}_2(\mathbf{x}^*) + \delta_{u_2}\right) + \delta_{s_u}^*.$$

This process allows the modeller to think separately about the model discrepancy for each sub-process modelled. We argue that it is advantageous to think about the model discrepancy for each physical process (and therefore intermediate variable), as these are the building blocks from which the simulator is built. By contrast, in thinking about the *external discrepancy*, the difference between the simulator output and the real world, one is thinking about the cumulative effect of many inadequacies (28). Furthermore, the intermediate variables will be active to different extents in the intermediate-output emulator, and this will show which discrepancies should be specified most carefully. Experiments such as those described by (11), where parameters are perturbed in order to learn the effects of the internal discrepancy, could be useful here, and we explore such an approach in subsection 4.5.1.

3.2. Adding intermediate variables. Whereas standard emulation demands no understanding of the simulator beyond its input and output variables, intermediate variable emulation requires some appreciation of a simulator's workings. This will involve study of the source code and literature, and liaising with experts where possible. Because of the importance of studying these intermediate variables, simulator code could be written in such a way that they are accessible. However, if this is not the case the code will need to be adapted to incorporate them.

For many simulators there will be more than one possible set of intermediate variables, and which works best may depend on the goal of emulation. It may be that one set of intermediate variables has a much simpler relationship to the input or output variables than the other, and might therefore add less information. Quantities calculated at each time step in the simulator can be a good choice. It is important to choose intermediate variables that give insight into the simulator's internal processes.

Sub-process	Intermediate variable	Time points	Intermediate variable elements
$A \to B$	$\mathbf{u}_{AB}(\cdot)$	1, 100, 314	$u^{1}_{AB}(\cdot)$, $u^{100}_{AB}(\cdot)$, $u^{314}_{AB}(\cdot)$
$B \to A$	$\mathbf{u}_{BA}(\cdot)$	25, 172	$u^{25}_{BA}(\cdot)$, $u^{172}_{BA}(\cdot)$

Table 3.1: A simulator contains two sub-processes, which are represented by two intermediate variables, $\mathbf{u}_{AB}(\cdot)$ and $\mathbf{u}_{BA}(\cdot)$. Each produces a time series, which we summarise by taking a small set of time points. Therefore $\mathbf{u}_{AB}(\cdot)$ is an intermediate variable with three elements, and $\mathbf{u}_{BA}(\cdot)$ has two elements.

The intermediate variables must separate the input variables from the output variables. That is, given the values of all the intermediate variables, information about the input variables should add no information about the output variables. There may occasionally be input variables that should be included as intermediate variables. This can happen when an input variable is involved 'near the end' of the simulator's calculations, for example if a quantity is multiplied by an input variable to produce an output variable. Simulator and system experts, papers and documentation can all provide insight into suitable choices of intermediate variables.

For many simulators, each input point produces intermediate variables with multiple values, often in a time series over a spatial region. In order to be able to emulate using these variables, they must be summarised. The summaries must contain enough information about the intermediate variables for the intermediate variable emulation model to capture the behaviour of the simulator adequately. They should not be too highly correlated with one another, and must reduce the dimension of the intermediate variable space enough to make emulation possible. Ideally, they should take a form that makes any analysis clear. One could approximate by an analytic function, using for example wavelets (2) or smoothing splines (10; 27), or by selecting variables at chosen time points, as in (6).

We therefore extend the notation used in section 2 to include the intermediate variables $\mathbf{u}_{\text{int}_1}(\cdot), \ldots, \mathbf{u}_{\text{int}_n}(\cdot)$, where $\mathbf{u}_{\text{int}_i}(\cdot)$ relates to the $i-th$ sub-process or intermediate variable (the label 'int' should be chosen to reflect the nature of the sub-process), and $u^{j}_{\text{int}}(\cdot)$ are the elements of $\mathbf{u}_{\text{int}}(\cdot)$. In the following sections, the term 'intermediate variable' will be used to refer to the set of elements (eg. time points of a time series) used to summarise a particular sub-process. As described above, this will usually be a vector, although it is not forced to be. For an example, see Table 3.1.

Note that the introduction of these intermediate variables does not lead to a greater number of model runs than a 'standard' emulation framework. The intermediate variable values are obtained 'for free' along with the output for each input point, assuming that the simulator has been set up appropriately.

3.3. Emulating Intermediate Variables. Instead of being used to model the simulator's output, as is usual practice, the input variables can be used to emulate the intermediate variables using the Gaussian Process model described in section 2. This could be viewed as one multivariate emulation problem, in which the same regression surface is used for every intermediate variable, as a set of multivariate emulation problems, one for each intermediate variable, or as a set of univariate emulation problems, one for each component of the dimension reduced intermediate variables. We take the second approach: each intermediate variable may have a different regression surface and correlation length, but within each intermediate variable, the quantities arising from the dimension reduction are jointly emulated using the same regression terms and correlation lengths. This will help in identifying the effects of inputs on the different sub-processes. We validate these emulators using careful diagnostics (1). In the example in section 4 most of the variation was accounted for by the regression surface, and diagnostics showed that using the same correlation length for each principal variable within an intermediate variable was not detrimental.

3.3.1. Sensitivity Analysis. To study the effects of the inputs on the intermediate variables, one can use sensitivity analysis methods (16; 14; 10). These techniques provide an opportunity for a model expert to judge whether the input parameters are having the intended effect on the sub-processes within the simulator.

This will generally involve studying the effect of the inputs on each element of the intermediate variables in turn. This would be of particular value where there are conflicting models, or during simulator development. Thus one goal of the input to intermediate part of this method is to better understand how the simulator is actually modelling the system, and to expose any surprisingly unrealistic behaviour. This insight provides more opportunities to adapt and improve the simulator.

3.3.2. Refining the input space. If system data are available for some intermediate variables, then *history matching* can be used. This method combines historical system observations z with simulator output $s(x)$ and an emulator $f(\mathbf{x})$ to refine the input space by ruling regions of input space out as 'implausible' (5; 32). It may be that some parts of the input space will lead to values of elements of intermediate variables that we believe to be impossible, even though the corresponding output may be quite realistic. This presents another opportunity to refine the input space. Emulators can be used to rule out regions of input space leading to values of $u^j_{int}(\cdot)$ outside a certain non-implausible interval. The value of $u^j_{int}(\cdot)$ at input point x is written $u^j_{int}(x)$.

Using the emulator, we can produce $(1 - p_\alpha)$ credible intervals for $u^j_{int}(\cdot)$ at any input point x. If this interval is completely outside of the non-implausible interval, the input point should be rejected. For the conclusions to be meaningful, the non-implausible interval should be chosen by experts, bearing in mind their beliefs about the simulator's accuracy. This is similar to the implausibility measure used in (32). History matching can be used to create a new, refined input design of approximately size n in the following way:

i Using a large dataset from the simulator, find the approximate fraction k of inputs which lead to this implausible behaviour
ii Generate a $(1 - k)^{-1} \times n$ point input design
iii Using an emulator, calculate $(1 - p_\alpha)$ credible intervals for $u^j_{int}(\cdot)$ over the input design
iv For each input point, if the interval is entirely outside the non-implausible interval, discard the point. Otherwise, keep it.

The simulator can now be run over this refined input design to produce training data, and an emulator can be built in the same way as before. As in any history matching process, this new emulator will only be valid on points contained within the refined input space.

This process can be repeated, each time creating a refined input space with enough points to build and validate an emulator in the new region. As the emulator's accuracy improves, more of the input space will be ruled out. If there is information about other intermediate variables this too can be introduced to further refine the space.

The advantage of generating a new design is that one has more control over the number and arrangement of points. Sometimes, however, computational costs will be an issue, or new runs will be unavailable. In this case, one can simply use the points from the original training data that are judged to be non-implausible, as in (35), augmented by as many additional runs as are feasible given the constraints of the problem.

If there is a clash between experts' knowledge about the state of the system (represented by the input values) and the model itself, this might manifest itself in a priori plausible areas of input space being ruled out. Depending on whether there was more confidence in the model or in the input values, this could lead to improvements in the simulator or its use.

Because the intermediate variables 'separate' the simulator inputs from the output variables, they contain the information to create any of the outputs. They also contain additional information about the model's internal behaviour. Therefore, the input to intermediate variable emulators can show which inputs are active in the simulator in a more comprehensive way than when considering only a selection of output variables.

3.4. Intermediate to Output. The next stage is to build an emulator from the intermediate variables to the output. Again we use the model structure in section 2, but now with the intermediate variables as emulator inputs. If the dimensions of the intermediate variable space have been reduced, the emulator will also need to incorporate a nugget term. Working with intermediate variables is made difficult because we are not

in control of the values they take. To produce training data one would ideally like to be able to choose points in intermediate variable space, and then run the simulator over these in the way that one can usually run it from the input variables. However, each point in intermediate variable space is associated with an input point, from which it was (at least notionally) created. There are potentially many invalid intermediate points, which a simulator is not capable of producing from any input point.

Because the intermediate variables represent processes that are linked to one another, and because the dimension reduced summaries of each intermediate variable will often come from very structured data over time and space, we can expect high correlation between the intermediate variables. This is in contrast to designs in input space, which we usually choose to be orthogonal and space-filling. This lack of orthogonality in the intermediate 'design' means that the coefficients of the regression parts of the emulator may be highly correlated and therefore not easily interpretable. So long as new input points are in the same region as the training data, the emulator will perform well. However if two intermediate variables are highly correlated in the training data, interpretations of the emulator's coefficients will be unreliable at new input points where this is not the case. It may be possible to use the input to intermediate emulators to generate a near-orthogonal design in intermediate space. However, this may either include infeasible points or omit feasible regions. Using the points given by the previous stage also preserves any refinement of the input space that has been done using constraints on the intermediate variables.

To ensure that non-orthogonality in the data is a result of the structure of the simulator, rather than poor design, the training data should ideally be large enough to cover the space satisfactorily, as is the case with any emulator. However, if the added structure built in by our use of the intermediate variables leads to simpler relationships, we may need a lower density of points.

We can gain some understanding of the simulator's behaviour by studying the correlations between the elements of the intermediate variables and the output. This can be done for each intermediate variable in turn to investigate the main effects, or using products of intermediate variables to show the effects of interactions on the output variables.

The first is simple to display, for example by plotting the correlations between each element of an intermediate variable and the output variable. The second can be displayed by creating a matrix whose $(i, j) - th$ entry is the correlation between the element-wise product of the i^{th} and j^{th} intermediate variables and the output, and plotting this matrix using a heatmap. We must be cautious in interpreting these values because of the non-orthogonality of intermediate variable space.

Now that both emulation stages have been done, they can be combined to make a two-stage emulator from input to output, and we can use this to generate samples from output space for new input data. This is done as follows:

i Generate an n-point input design that is in keeping with the training data's non-orthogonality.
ii For each input point, generate a sample of points from the intermediate variable space using the input to intermediate emulators
iii For each of these sampled intermediate variable points, generate a random value from the output distribution using the intermediate to output emulator

In some cases, for example ones similar to those set up in subsection 3.1, this may outperform a standard emulator.

3.4.1. Internal discrepancy. It is at this stage that we can perform an analysis on the internal discrepancy. By perturbing the values of the elements of the intermediate variables, we can create mini designs with variation in just one intermediate variable. We can then run these sets of points through the emulator[1] and measure the effect on the output. Gauging the appropriate magnitude of the perturbation is a job for a field expert. In (28) Strong et. al suggest asking "If we knew true values of the inputs to this [sub-process], would we believe

[1]It would be preferable to 'part run' the simulator given a design in intermediate variable space, however it is unlikely to be possible.

with certainty that the [intermediate variable] would equal the true value of the corresponding intermediate [sub-process]?". If the answer is 'no', then we can introduce a discrepancy term to this intermediate variable. One natural way to do this is with normally distributed noise, possibly correlated between the elements of the intermediate variable. Thinking about the discrepancy between each of these sub-processes and the system behaviour it models seems a useful and more intuitive alternative to thinking just about the external discrepancy. Furthermore, as we saw with error terms in subsection 3.1, our beliefs about the external discrepancy in relation to these discrepancies on intermediate variables will depend on how each intermediate variable relates to the output. We give an example of this in subsection 4.5.1.

3.5. Validating the intermediate variable emulator. Because the intermediate variables (before dimension reduction) should capture all the information in the simulator at each time point, it should be possible to use the dimension-reduced intermediate variables to emulate the output. This is a good test of the choice and implementation of the intermediate variables and their dimension-reduction. The emulator should also be able to make accurate predictions from intermediate to output variables for other datasets. If it isn't, this is a sign that at some point in the process information has been lost, either in the dimension reduction stage or more fundamentally while selecting the intermediate variables, or in the structure of the input design.

4. Example: an ocean carbon cycle model.

4.1. Compartmental Models. Compartmental models are a significant subclass of computer simulators which fit well into the intermediate variable emulation framework. These models work by monitoring the transmission of a tracer (eg. energy or some substance) between different *compartments* in the system. If the model is 'closed', then the total amount of energy or substance remains the same, and must be accounted for by the levels in the compartments at all times. If the model is 'open' then the total amount is allowed to change over time. For example, the parts of a patient's body may be represented as compartments, and the flow of a particular drug between them could be modelled. Compartmental models are used in many fields, for example medicine (34), epidemiology (36), agriculture (30), and oceanography (13; 18), as in the example in this paper. In general one can use the concentrations of the tracer(s) in each compartment or the flow of tracers between compartments as intermediate variables.

4.2. OG99NPZD. OG99NPZD (18) is a compartmental ocean carbon cycle simulator based on the nitrogen-based ecosystem model developed by Fasham et. al. (8). Its input variables are shown in Table 4.1. Ocean carbon cycle simulators are incorporated into general circulation models (8), and capture some globally important phenomena, most notably the 'spring bloom' of phytoplankton in the North Atlantic. They are interesting candidates for Bayesian emulation because there is wide uncertainty around the model itself, with competing physical and biological models (see for example (21) for a different simulator of the same system). There is also uncertainty around the appropriate values of the input parameters (26). A standard input to output emulator with only first order terms in the regression surface captured the behaviour well, with the correlated error term accounting for little of the variation. However, although there are no real gains to be made in better capturing complicated behaviour, we can still use intermediate variable emulation to gain insight that would not otherwise be available to us.

OG99NPZD tracks the concentrations of nitrogen in four compartments: nutrient (N), phytoplankton (P), zooplankton (Z) and detritus (D) at multiple depth levels. At each depth level, the 'source minus sink'

OG99NPZD parameter	Description
a	Maximum photosynthetic rate at temp $= 0°C$ (day^{-1})
c	Max. photosynthesis - variation of temperature factor exponent ($(°C)^{-1}$)
μ_{PP}	Conc. dependent phytoplankton specific mortality ($d^{-1}(mmol\ N\ m^{-3})^{-1}$)
μ_P	Specific phytoplankton mortality rate (day^{-1})
μ_{ZZ}	Conc. dependent zooplankton specific mortality ($d^{-1}(mmol\ N\ m^{-3})^{-1}$)
γ_1	Zooplankton assimilation efficiency phytoplankton
g	Maximum grazing rate (d^{-1})
ϵ	Prey capture rate ($d^{-1}(mmol\ N\ m^{-3})^{-n}$)
γ_2	Excretion rate ($days^{-1}$)
P_{AR}	Ratio of P_{AR} to total downwelling solar irradiance at sea surface
c_{PP}	Ratio of chlorophyll to total pigment
α	Initial slope of photosynthesis v irradiance curve ($(mg\ C\ (mg\ Chl)^{-1}\ (E\ m^{-2})^{-1}$)
K_1	Half-saturation conc. for nutrient uptake ($mmol\ N\ m^{-3}$)
η	Phytoplankton specific respiration (d^{-1})
w_s	Detritus sinking velocity ($m\ day^{-1}$)
μ_D	Remineralisation rate (day^{-1})

Table 4.1: Input parameters for OG99NPZD with descriptions.

equations (4.1) to (4.4) determine the transfer of nitrogen between these compartments at each time step.

$$\text{sms}(P) = \underbrace{\bar{J}(z,t,N)\,P}_{\text{growth}} - \underbrace{G(P)\,Z}_{\text{grazing}} - \underbrace{\left(\mu_P P + \mu_{PP} P^2\right)}_{\text{death}} \tag{4.1}$$

$$\text{sms}(Z) = \underbrace{\gamma_1 G(P) Z}_{\text{grazing}} - \underbrace{\gamma_2 Z}_{\text{excretion}} - \underbrace{\mu_Z Z^2}_{\text{mortality}} \tag{4.2}$$

$$\text{sms}(D) = \underbrace{(1-\gamma_1)\,G(P)Z}_{\text{unassimilated food}} + \underbrace{\mu_{PP} P^2}_{\text{dead P}} + \underbrace{\mu_Z Z^2}_{\text{dead Z}} - \underbrace{\mu_D D}_{\text{remineralisation}} - \underbrace{w_s \frac{\partial D}{\partial z}}_{\text{sinking}} \tag{4.3}$$

$$\text{sms}(N) = \underbrace{\mu_D D}_{\text{remineralisation}} + \underbrace{\gamma_2 Z}_{\text{excretion}} + \underbrace{\mu_P P}_{\text{dead P}} - \underbrace{\bar{J}(z,t,N)\,P}_{\text{P growth}}. \tag{4.4}$$

Figure 4.1 and Table 4.2 depict the transfers of nitrogen between compartments in OG99NPZD, along with the physical process linked to each. This is a highly idealised set-up intended for demonstrating the technique and does not include all processes typically considered in a real oceanographic application. In particular, vertical transport by water movements due to the physical processes of advection, diffusion and mixing are not considered. The only exchange of material between levels is due to the sinking of detritus.

At each time step, the source-minus-sink equations are used to update the concentrations of nitrogen in each compartment. We therefore use the flux of nitrogen between each pair of compartments at each time step as our intermediate variables, since this is how the biological processes are modelled. The full set of intermediate variables, along with the biological processes represented, are given in Table 4.2. Throughout the example we will use italics for input variables, the $\mathbf{u}_{\text{int}}(\cdot)$ notation for intermediate variables, and capital letters for output variables.

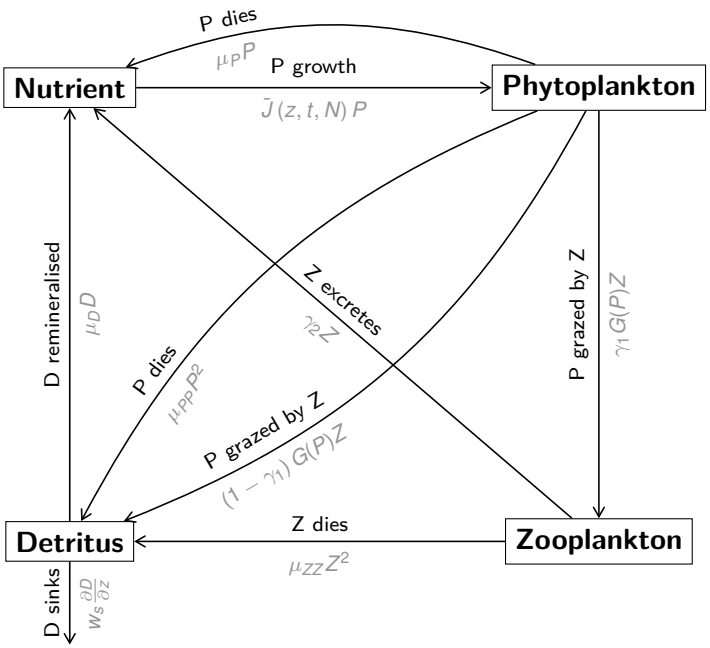


Figure 4.1: Transfers of nitrogen modelled by the Oschlies-Garçon simulator

Transfer	Processes in OG99NPZD	Variable name
N to P	P growth	$\mathbf{u}_{np}(\cdot)$
P to N	P mortality	$\mathbf{u}_{pn}(\cdot)$
P to Z	P grazed	$\mathbf{u}_{pz}(\cdot)$
P to D	P mortality, P grazed	$\mathbf{u}_{pd}(\cdot)$
Z to N	Z excretion	$\mathbf{u}_{zn}(\cdot)$
Z to D	Z mortality	$\mathbf{u}_{zd}(\cdot)$
D to N	D remineralised	$\mathbf{u}_{dn}(\cdot)$
D lost	D sinks	$\mathbf{u}_{ds}(\cdot)$

Table 4.2: Intermediate variables in terms of nitrogen transfer for OG99NPZD, and the system processes that contribute.

4.3. Data. Two data sets, 'OG100' and 'OG1000', were created using 100 and 1,000 point Latin hypercube designs (LHDs) over OG99NPZD's input space, respectively. We checked that they were roughly orthogonal and had good space-filling properties. The outputs of OG99NPZD are given as a time series over 57 depth levels, but can be 'depth-integrated'. This is effectively an average over the depth levels, and reduces the variable to a single time series. We used the log of depth-integrated particulate organic nitrogen (`PON`) at two time points (chosen by principal variables) as our overall output. `PON` is the sum of the phytoplankton, zooplankton and detritus concentrations. We can also depth-integrate the intermediate variables as the first step of dimension reduction.

We will name the depth-integrated intermediate variables according to the nitrogen transfer taking place, for example $\mathbf{u}_{np}(\cdot)$ is the transfer from nutrient to phytoplankton [2]. The Box-Cox procedure suggested that

[2]The exception to this rule is $\mathbf{u}_{ds}(\cdot)$, which represents detrital sinking, where detritus is simply lost from the model as it sinks

Variable	Time points	Variance Explained	Names
$\mathbf{u}_{np}(\cdot)$	303, 1, 2	0.991	u_{np}^{303}, u_{np}^{1}, u_{np}^{2}
$\mathbf{u}_{pn}(\cdot)$	110, 24	0.993	u_{pn}^{110}, u_{pn}^{24}
$\mathbf{u}_{pz}(\cdot)$	172, 1, 365, 40	0.997	u_{pz}^{172}, u_{pz}^{1}, u_{pz}^{365}, u_{pz}^{40}
$\mathbf{u}_{pd}(\cdot)$	109, 1	0.990	u_{pd}^{109}, u_{pd}^{1}
$\mathbf{u}_{zn}(\cdot)$	140, 21	0.994	u_{zn}^{140}, u_{zn}^{21}
$\mathbf{u}_{zd}(\cdot)$	124, 1	0.992	u_{zd}^{124}, u_{zd}^{1}
$\mathbf{u}_{dn}(\cdot)$	121, 2, 14	0.995	u_{dn}^{121}, u_{dn}^{2}, u_{dn}^{14}
$\mathbf{u}_{ds}(\cdot)$	118, 1	0.992	u_{ds}^{118}, u_{ds}^{1}

Table 4.3: Principal variables chosen using OG1000 and stipulating that 99% of variance should be explained.

each of the intermediate variables involving zooplankton ($\mathbf{u}_{pz}(\cdot)$, $\mathbf{u}_{zn}(\cdot)$ and $\mathbf{u}_{zd}(\cdot)$) suited a log transform, and this was used throughout. As in subsection 3.2 we will use superscripts to specify the time point elements of each intermediate variable, so for example $u_{np}^{24}(\cdot)$ is the depth-integrated nutrient to phytoplankton transfer at time 24. To reduce the dimension of the data, we used principal variables, as described by Cumming and Wooff (6).

Principal variables is a dimension reduction technique in which only a relatively small number of variables are kept, chosen so that they maximise the information retained (15). In the iterative method proposed by (6), those variables chosen are likely to be those that would have high loadings on important principal components. In contrast to many dimension reduction techniques, the principal variables method enables us to discard the rest of the data. We also believe that continuing to deal with individual variables should make our model more interpretable, and would also make it simpler to think about quantities such as model discrepancy. Table 4.3 shows the principal variables found by stipulating that 99% of the variance should be explained. We will refer to the dataset containing the inputs of OG1000 and these selected principal variables of the intermediate and output variables as 'OGPV'.

Throughout the analysis we must bear in mind the correlations between the intermediate variables. We found that later time points of the zooplankton related transfers were highly correlated, and $\mathbf{u}_{dn}(\cdot)$ and $\mathbf{u}_{pd}(\cdot)$ were highly correlated. Figure 4.2 shows a heat map of the correlation matrix of OGPV.

below the depth range covered.

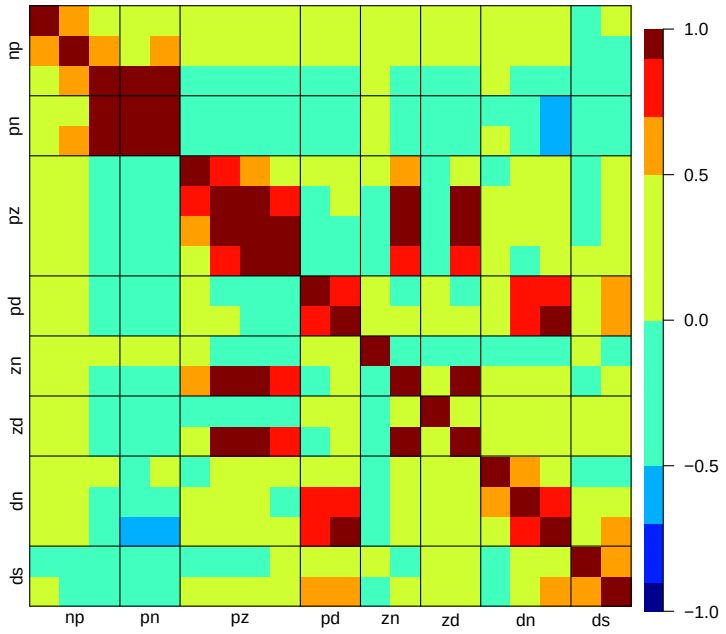


Figure 4.2: Heat map of the correlation matrix for principal variables from OGPV. Each sub-interval represents a particular principal variable. Within each intermediate variable they are ordered by time (from left to right and from top to bottom).

4.4. Emulating Intermediate Variables. The first emulator we built used the input and intermediate variables in OGPV as training data. To understand the most active variables and pairwise interactions in the intermediate variables, a second order surface was chosen for each intermediate variable using stepwise model selection with the Bayesian information criterion (BIC), and one correlation length was estimated per intermediate variable. For sensitivity analysis, we consider two measures from (16). The *main effect* of input x_i, given by

$$z_i(x_i) = \mathrm{E}(Y \mid x_i) - \mathrm{E}(Y),$$

and the variance based measure

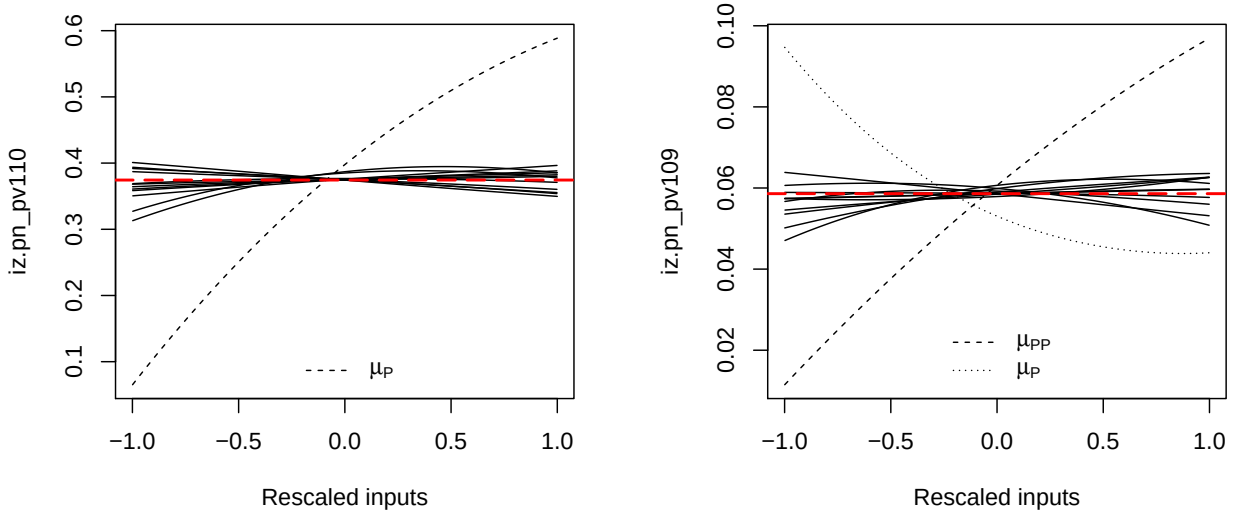
$$V_p = \mathrm{var}\left[\mathrm{E}(Y \mid \mathbf{X}_p)\right],$$

where $\mathbf{x}_p$ is a subgroup of one or two input variables. These quantities therefore each give a single numeric summary of the importance of an input or pair of inputs.

The plots in Figure 4.3 give some examples of main effects for particular intermediate variables, and Table 4.4 shows the proportion of variance explained by each input variable for each intermediate variable. We found that $\mathbf{u}_{\mathrm{np}}(\cdot)$ and $\mathbf{u}_{\mathrm{pn}}(\cdot)$ were most affected by μ_P, the zooplankton-related intermediates by γ_2, and to a lesser extent by γ_1 and ϵ, $\mathbf{u}_{\mathrm{pd}}(\cdot)$ and $\mathbf{u}_{\mathrm{dn}}(\cdot)$ were mostly governed by μ_{PP} and a little by μ_P, and $\mathbf{u}_{\mathrm{ds}}(\cdot)$ was most affected by ω_s and μ_{PP}. The proportions of variances explained by each input showed that interaction terms were relatively unimportant for all intermediates. An input to output emulator showed that the output variable on which we chose to focus, `PON`, is governed almost entirely by μ_P. So, clearly μ_P is an active input, but some others, for example μ_D or ω_s, although not active for `PON`, are active in some parts of the model.

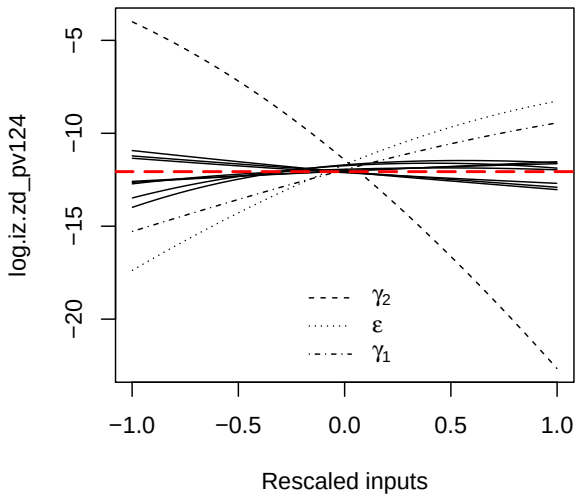
Intermediate	Time	α	a	γ_1	c	ω_s	ϵ	g	K_1	μ_P	P_{AR}	C_{PP}	γ_2	μ_{PP}	μ_{ZZ}	η	μ_D
$\mathbf{u}_{np}(\cdot)$	1		0.42		0.24				0.33								
	2		0.18		0.16					0.16							0.05
	303									0.86							
$\mathbf{u}_{pn}(\cdot)$	24									0.94							
	110									0.87							
$\mathbf{u}_{pz}(\cdot)$	1			0.23			0.73										
	40			0.18			0.40						0.33				
	172			0.12			0.23						0.57				
	365			0.10			0.18						0.53				
$\mathbf{u}_{pd}(\cdot)$	1													0.93			
	109									0.20				0.59			
$\mathbf{u}_{zn}(\cdot)$	21			0.06			0.06						0.63				
	124			0.11			0.20						0.59				
$\mathbf{u}_{zd}(\cdot)$	1														0.88		
	124			0.08			0.16						0.67				
$\mathbf{u}_{dn}(\cdot)$	2													0.13			0.86
	14									0.07				0.64			0.28
	121									0.21				0.52			0.06
$\mathbf{u}_{ds}(\cdot)$	1					1.0											
	118					0.36								0.19			
PON	9									0.94							
	106									0.84							

Table 4.4: Proportions of variance explained by each input variable for each intermediate variable for the emulator built with OG1000. For clarity, only values above 0.05 are shown.

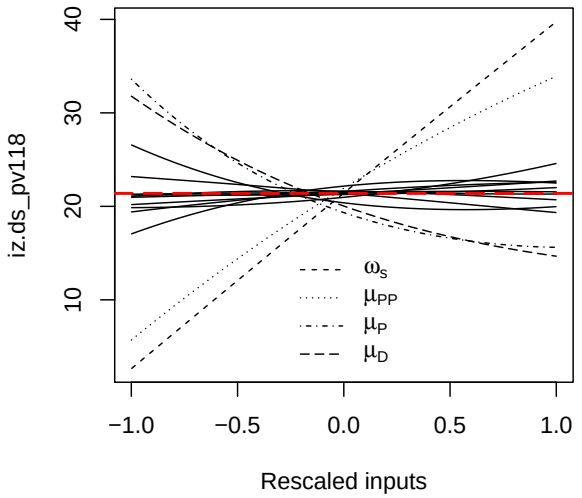


(a) $\mathbf{u}_{\text{pn}}(\cdot)$, time 110

(b) $\mathbf{u}_{\text{pd}}(\cdot)$, time 109



(c) $\mathbf{u}_{\text{zd}}(\cdot)$, time 124



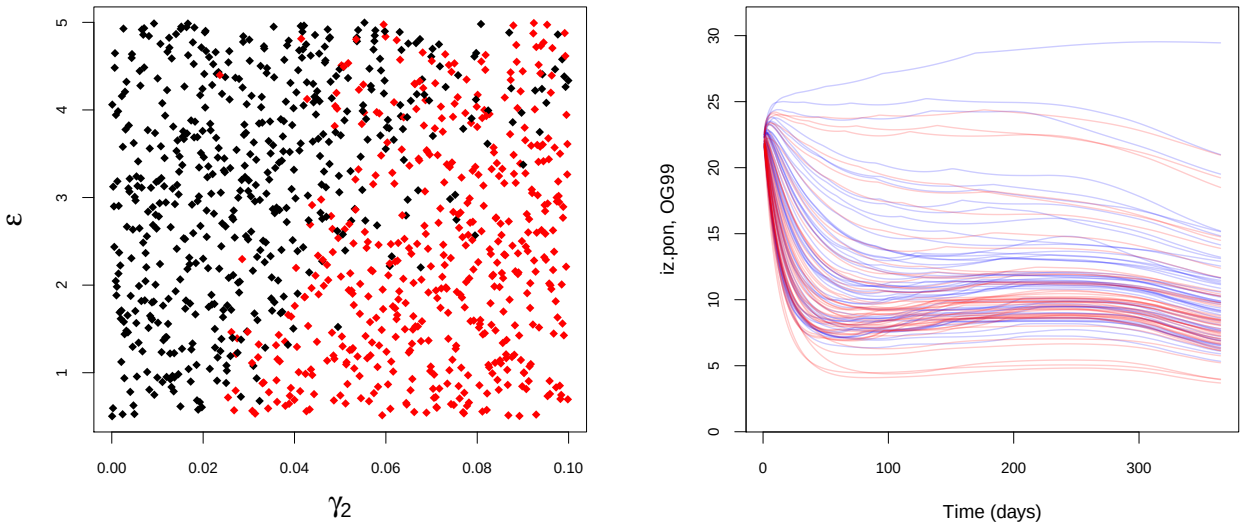
(d) $\mathbf{u}_{\text{ds}}(\cdot)$, time 118

Figure 4.3: Main effects plots for some of the intermediate variables in OG99. Only the most active inputs are distinguished. Each line shows the expected value of the intermediate variable as a particular input variable changes. The red dashed line is at zero.

Constraining input parameter values is extremely difficult for some ocean ecosystem parameters (26), and so discerning which input parameters are less active can inform where to focus effort.

4.4.1. Unrealistic behaviour. A key purpose of this step is to detect unrealistic behaviour in the sub-processes, and potentially to use this to rule out portions of input space. It turns out that for large regions of our input design the nitrogen transfers involving zooplankton can rapidly reach a very small value, implying zooplankton extinction. This is the sort of phenomenon we may wish to rule out. This is similar to the 'iterative refocussing' approach in (35), where parts of the input space are ruled out that almost certainly could not reproduce the desired output.

Although these simulator runs have unrealistic zooplankton related intermediates, their vertically integrated



(a) Pairs of OG1000 inputs plotted against one another, coloured according to the latest time points of the zooplankton related intermediates. If any is below 10^{-8}, the point is red, otherwise it is black.

(b) Time series of PON from OG100, coloured according to whether zooplankton appears to become extinct (red) or not (blue). Note that there are many input points leading to zooplankton extinction that have a plausible output value.

Figure 4.4: Some of the behaviour of OG99NPZD in terms of zooplankton extinction.

PON values have a similar distribution to those with more realistic zooplankton behaviour, as demonstrated by the plots in Figure 4.4b. This is a somewhat artificial example, since one could alternatively plot the zooplankton concentration. However, other types of unusual behaviour in internal processes could be missed by focussing only on concentration variables, since not all potentially important behaviour can be captured in these variables (e.g. unusual behaviour in growth and mortality might cancel out). Intermediate variable emulation provides an opportunity to look more closely at these processes, and potentially to refine the underlying model.

Figure 4.4 shows some clear links between input variables and a tendency towards zooplankton extinction in OG99NPZD. The pairs plots involving other variables mostly show very little pattern. This is not surprising given the results of the sensitivity analysis.

Using the strategy outlined in subsection 3.3.2, the emulators enable us to rule out portions of the input space leading to undesirable values of zooplankton-related intermediate variables. Although in reality the values of p_α (the confidence level of the interval) and $L_{\texttt{int}}$ (the lower limit of the acceptable interval for intermediate variable $\mathbf{u}_{\text{int}}(\cdot)$) should be chosen carefully by an expert, for the purposes of this example we chose $p_\alpha = 0.05$ and $L_{\texttt{int}} = 10^{-6}$ for the latest time point of each of the zooplankton-related intermediates. Because the focus is on values that are too low, the upper limit of the interval is effectively infinite.

OGPV gave $k_{OG} = 0.478$. To generate a new input design containing roughly 1,000 points, we therefore included $1,916$ points in the initial design. The emulator built from the unrefined dataset OGPV was run over this input design, and used to create 95% credible intervals for each intermediate variable at each input point.

In this example, the emulators for zooplankton related intermediate variables all emulate the logarithm of the transfer, and so intervals are built on the log-scale, and the bounds then transformed back to the original scale. This has the advantage that the lower bounds are all above zero, reflecting the simulator's inability to

produce negative values. Using the emulator built from OGPV, any point in the input space can be categorised as either being very likely to result in zooplankton extinction, or not being likely enough to disregard.

In this way, points from the input design that the emulator judged likely to tend towards zooplankton extinction were ruled out, reducing the number of points to 1004. We can now re-run the simulator over this refined set of input points to form training data, build new emulators and re-do the intermediate variable analysis up to this point. As mentioned in subsection 3.3.2, if computational expense or availability of simulator runs is an issue, we could alternatively use the points from OGPV that are judged to be in non-implausible space. The sensitivity analysis may become less transparent, since the refined input space will not necessarily be near-orthogonal, but one can still learn about which inputs (or groups of inputs) are driving which sub-processes. Some example plots of the refined input space are shown in Figure 4.5, and we see that a distinct area of the original space has been removed. These are some of the inputs most influential in the zooplankton transfers, as is clear from Table 4.4.

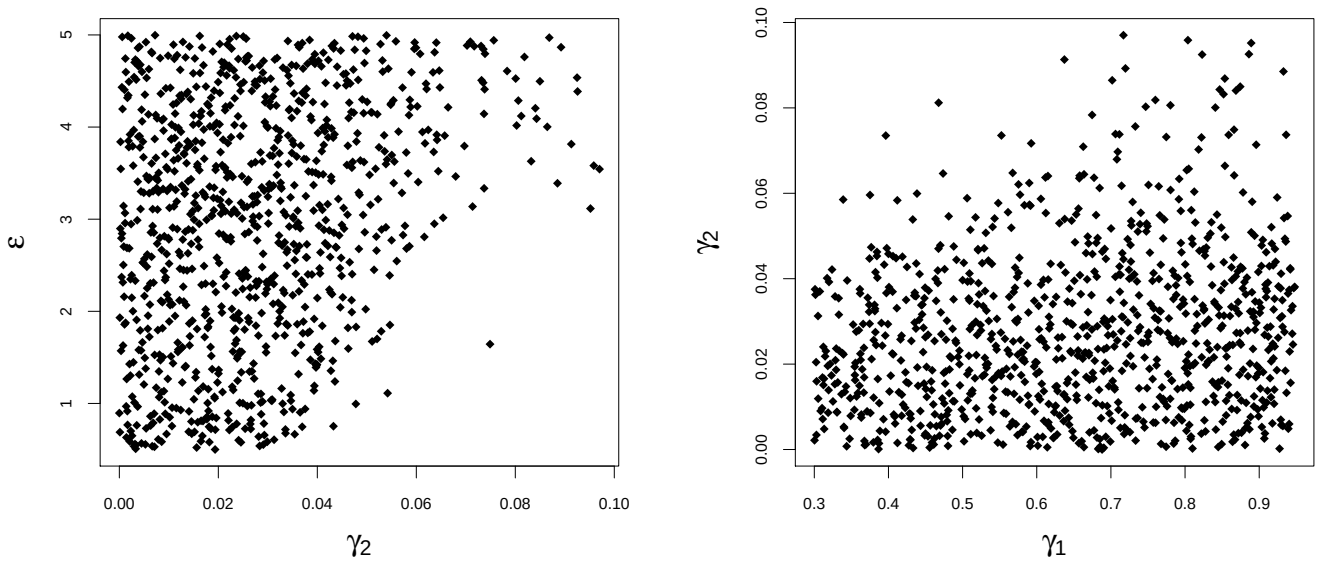


Figure 4.5: The pairwise distribution of input points in the refined input design, for two pairs of input variables.

Figure 4.6 compares the distributions of the latest principal variables for two of the zooplankton transfers. In both cases, the distribution in the refined space (the bottom two plots) has a greater concentration of higher values. Interestingly, in both cases there also seems to be a reduction in the highest values as well as the lowest. If this process were repeated the input space could be refined further, but for the purposes of illustration we have performed one wave.

This stage has increased our understanding of how each input variable contributes to the representation of the ocean carbon cycle in OG99NPZD. The conclusions drawn are independent of the choice of output variable, and so can be used to count certain input parameters as less active in general than others. That is, because the intermediate variables contain all information necessary to know the output variables, if an input variable is inactive for all intermediate variables then it cannot be active for any output variables. Furthermore the intermediate variables capture more of the behaviour of the sub-processes, so internal activity that may not be visible in a typical set of model outputs has been taken into account in the analysis.

We were also able to reduce the input space using unrealistic intermediate variable values, in a way that

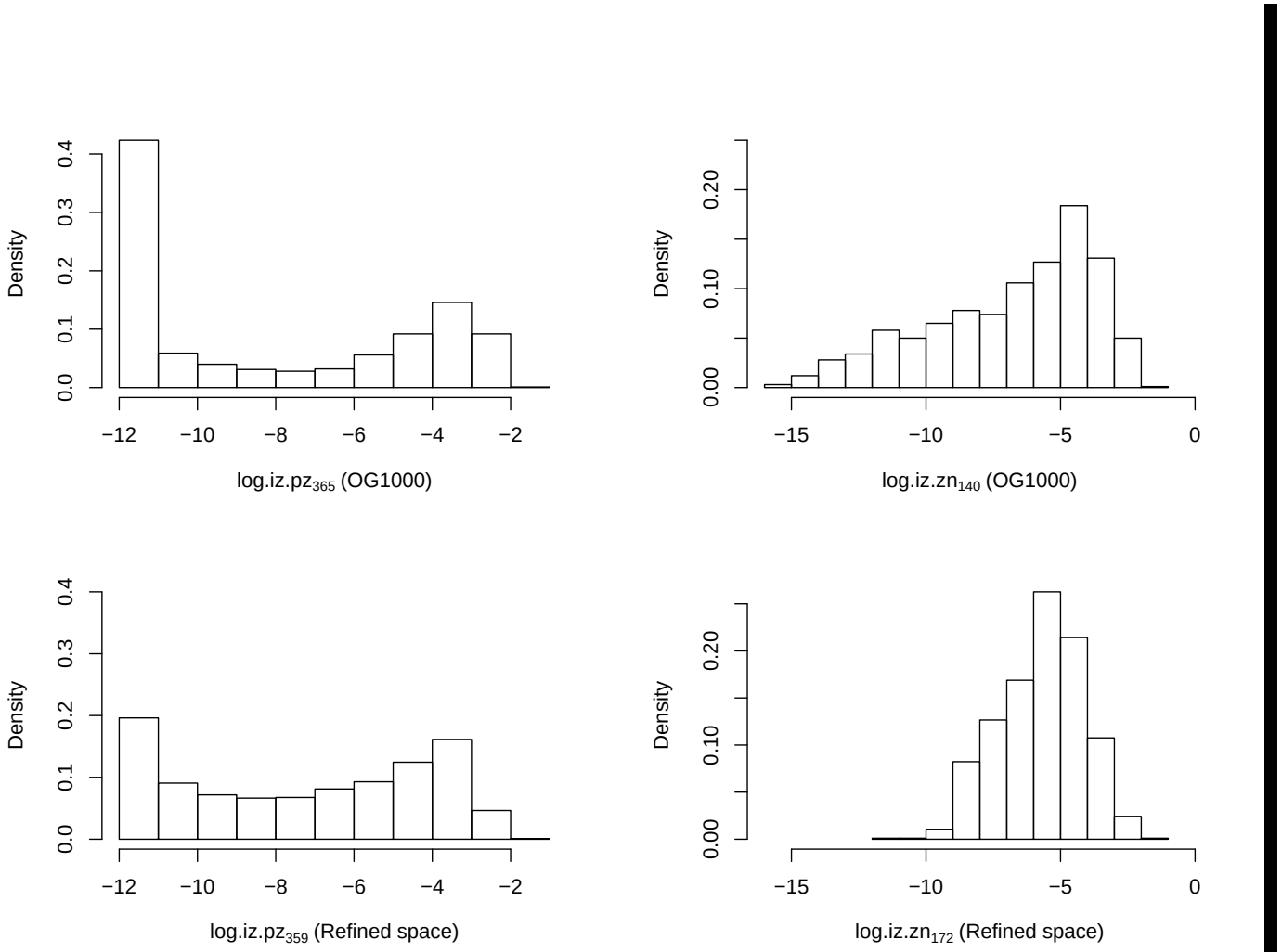


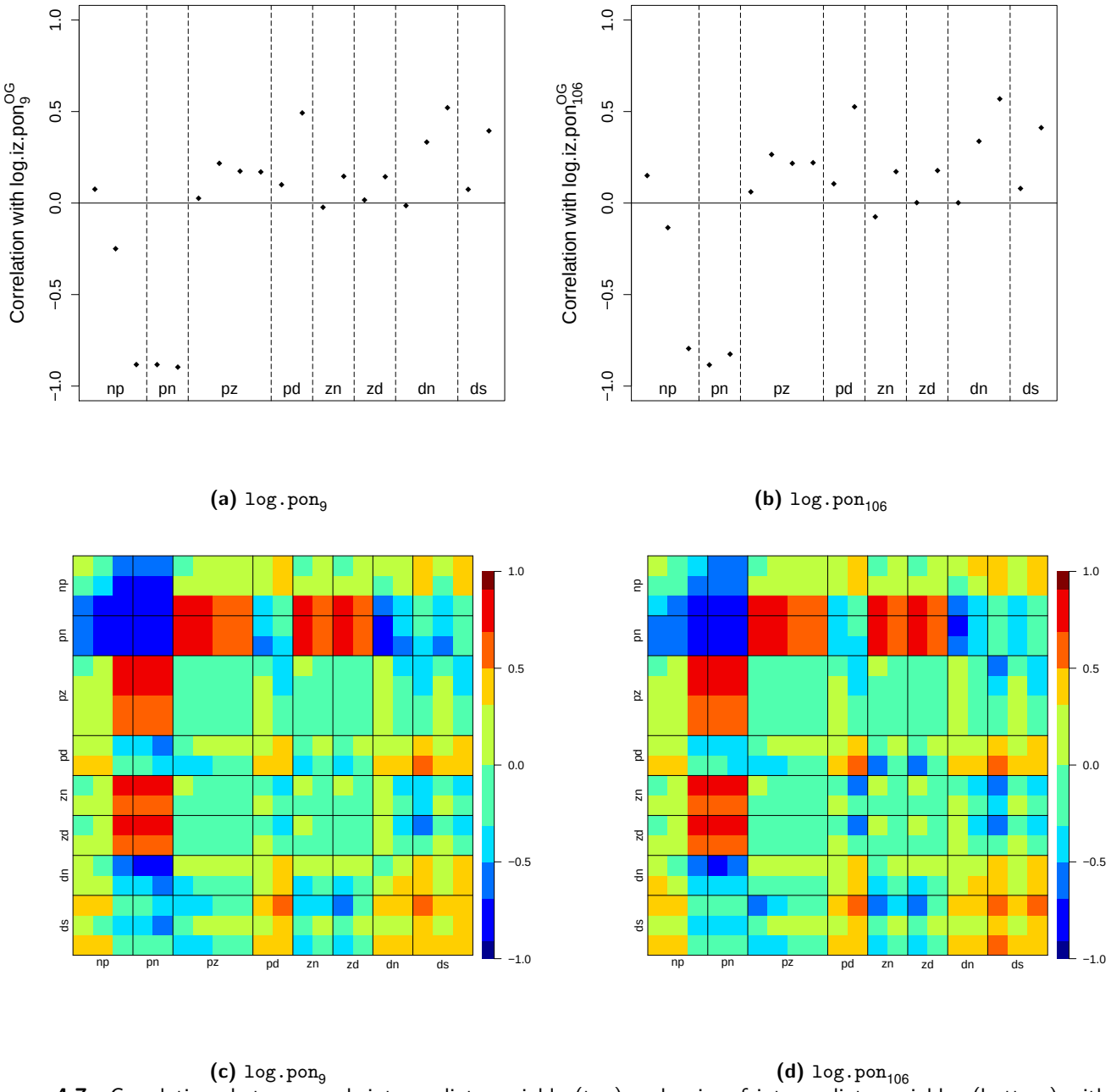
Figure 4.6: Distributions of the principal variable with the latest time point of some zooplankton transfers for OG1000 and for the refined input space. The left-hand two plots show the latest time points of $\mathbf{u}_{\text{pz}}(\cdot)$, the right-hand two show the latest time poits for $\mathbf{u}_{\text{zn}}(\cdot)$. The top two show OG1000 and the bottom two show the refined space. Note that the principal variable time points have been re-selected for the refined space.

would not be possible using PON output of our model alone, leading to a new input design in which there is less of a tendency towards zooplankton extinction.

4.5. Example: Intermediate to Output. Figure 4.7 shows the correlations between the intermediate variables and the output for OG1000. We see that $\mathbf{u}_{\text{pn}}(\cdot)$ and the last time step of $\mathbf{u}_{\text{np}}(\cdot)$ are all negatively correlated with PON. They are strongly correlated with one another (see Figure 4.2) and so without further investigation we can't tell which process is responsible. There is also a suggestion of interaction effects between $\mathbf{u}_{\text{pn}}(\cdot)$ and some of the zooplankton related intermediate variables, but it is difficult to tell whether this is just a side effect of the influence of $\mathbf{u}_{\text{pn}}(\cdot)$.

The intermediate variables are all positive on their original scales, but the logarithms of the zooplankton related intermediates, which are the quantities used to produce these plots, are negative. This affects the signs of correlations between products of intermediate variables when one is zooplankton related.

This stage of intermediate variable emulation could help an expert to isolate the important parts of their



(a) $\texttt{log.pon}_9$ **(b)** $\texttt{log.pon}_{106}$

(c) $\texttt{log.pon}_9$ **(d)** $\texttt{log.pon}_{106}$

Figure 4.7: Correlations between each intermediate variable (top) and pairs of intermediate variables (bottom) with each output variable from OG1000. Each set of principal variables is in time order.

model. For example here, it would seem to be beneficial to concentrate on $\mathbf{u}_{\text{pn}}(\cdot)$ and the modelling that goes into calculating it in order to improve OG99NPZD.

4.5.1. Internal discrepancy analysis. Using the intermediate to output emulator, we can perform an internal discrepancy analysis, as in subsection 3.4.1. In this example, we have little information about experts' beliefs in the simulator, and therefore made choices that seemed general and reasonable. This analysis was

performed on a subset of the original data OG1000, containing 100 points.

For each intermediate variable our discrepancy terms have a multivariate normal distribution, with the standard deviation for each element being 10% of the range of values in the training data. Within each intermediate variable, the correlation of the discrepancy term between the different time points was 0.9. For each intermediate variable $\mathbf{u}_{\text{int}}(\cdot)$ the procedure was as follows:

1. For each of the 100 points in the dataset, generate a sample of n discrepancy vectors from the multivariate normal distribution described above.
2. Add these discrepancy vectors to the intermediate variable values of the input point, creating n new input points where $\mathbf{u}_{\text{int}}(\cdot)$ varies but all other intermediate variables are constant.
3. Run these n points through the intermediate-output emulator, generating n output distributions.

This procedure yielded, for each intermediate variable, 100 sets of n output distributions, each corresponding to a different point in the general intermediate variable space. These can be used to gauge the effect of uncertainty about each sub-process on the simulator output.

Figure 4.8 shows the results of this procedure for $u_{\text{pn}}^{113}(\cdot)$ and $u_{\text{dn}}^{184}(\cdot)$. It is clear that the effect on PON of adding the internal discrepancy to $\mathbf{u}_{\text{pn}}(\cdot)$ is much greater than for $\mathbf{u}_{\text{dn}}(\cdot)$. There also appears to be a trend in the amount of variation in the output for $u_{\text{pn}}^{113}(\cdot)$, with perturbations around lower values of $u_{\text{pn}}^{113}(\cdot)$ leading to more variation in PON than those around higher values. These trends could be explored further, for example by emulating the variance of the discrepancy across the intermediate variable space.

To gain a more general overview of the effect of the internal discrepancy relating to each intermediate variable, we can plot histograms of the variances of the PON at each point, as in Figure 4.9. Because the discrepancies were generated jointly around the point in intermediate space, these now relate to intermediate variables, rather than just single elements. It is clear that, across the intermediate space, uncertainty about $\mathbf{u}_{\text{pn}}(\cdot)$ is the most influential, and that uncertainty in $\mathbf{u}_{\text{dn}}(\cdot)$ and $\mathbf{u}_{\text{pd}}(\cdot)$ appears to matter very little. This is perhaps unsurprising, since we saw in Figure 4.7 that both time points of $\mathbf{u}_{\text{pn}}(\cdot)$ are very active in the intermediate to output emulator. This further reinforces the point made in subsection 3.4.1, that to understand the discrepancy in the output, it would be important to understand the discrepancy in the more active intermediate variables, such as $\mathbf{u}_{\text{pn}}(\cdot)$.

4.5.2. Combined emulator. The intermediate to output emulator is of use for verifying the process by putting the two stages together, as in subsection 3.4. Figure 4.10 shows boxplots (the blue boxplots, on the left of each pair) of these samples with the true output time series for reference. Even in this simple case, where the relationship between intermediate and output variables was fairly simple and linear, the Intermediate Variable Emulation structure suggests a more careful form of emulation which respects the key uncertainties in the problem. Furthermore, since we can incorporate internal discrepancies, as we have in subsection 4.5.1, use of the internal variables gives us better emulation, not just of the function itself but (arguably more importantly) of the real world process that the function is intended to represent.

5. Summary. We have presented Intermediate Variable Emulation, a new method for developing understanding of a simulator using Bayesian emulation, and for making complicated simulators more tractable. Our method represents a new approach, by involving the internal workings of the simulator in the emulation process. Whereas in the standard emulation framework the simulator is treated as a black box, here we make use of the sub-processes it models, to form what we name 'intermediate variables'. As this information is usually readily available, this seems a natural step. While this has an initial cost as the intermediate variables are set up, the benefits they bring are significant. Core to our method is the belief that good coverage of information on internal processes is key to obtaining good insight. For some simulators, setting up these intermediate variables may seem prohibitively difficult, even if conceptually their meaning is clear. However, it would be within the gift of computer modellers to think in terms of intermediate variables, and to make them easily accessible as part of the simulator. Therefore this paper should be seen in some ways as a manifesto for doing just that. We see this as analogous to the paradigm shift that led to the field of experimental design, where scientists

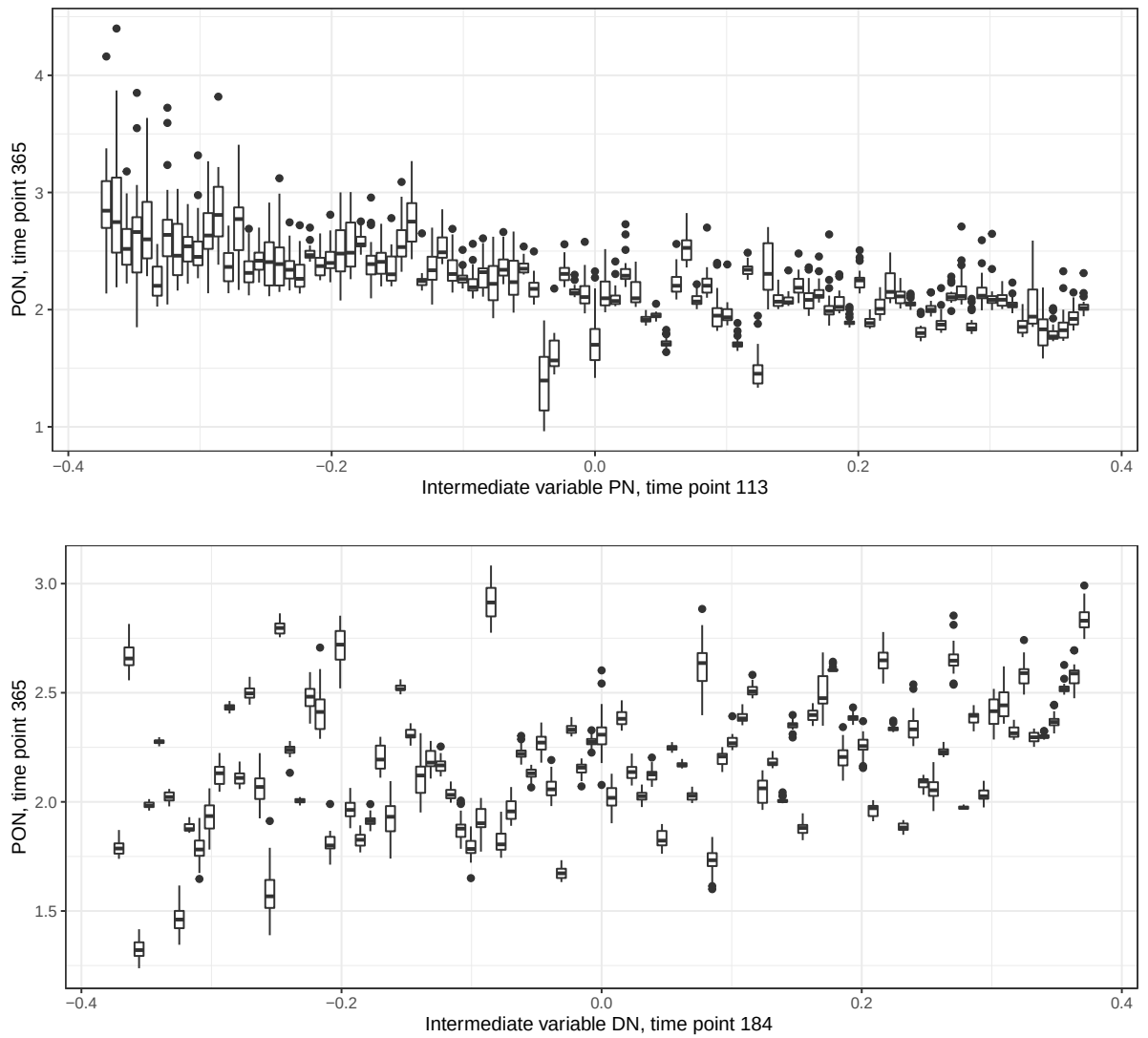


Figure 4.8: Boxplots showing the output distribution for the perturbed values of two time points of intermediate variables, $u_{pn}^{113}(\cdot)$ and $u_{dn}^{184}(\cdot)$. Each boxplot summarises the variation induced by the discrepancy sample around a single point.

consider the analysis they plan to do with their data and plan their experiment accordingly.

As well as a generally improved understanding of the simulator's behaviour, intermediate variable emulation brings several practical advantages. With the help of an expert, intermediate variable emulation can be used to comprehensively determine which inputs to the simulator are active, and therefore where to focus effort on understanding the system. The method can also be used to refine input space and potentially find specific areas for simulator improvement, through the analysis of unrealistic behaviour in intermediate variables. This may lead to history matching or similar, in which parts of the input space leading to undesirable behaviour are ruled out, or it may lead to re-modelling of a particular sub-process. The intermediate to output emulator in turn shows which internal processes are driving each simulator output.

The intermediate variable emulation approach provides a more intuitive and flexible structure for specifying model discrepancy, since the discrepancies for each internal process can be considered separately. We have also shown an example where the complicated nature of the simulator means that this structure can capture more of the variation, in a more natural way, than emulating from input to output.

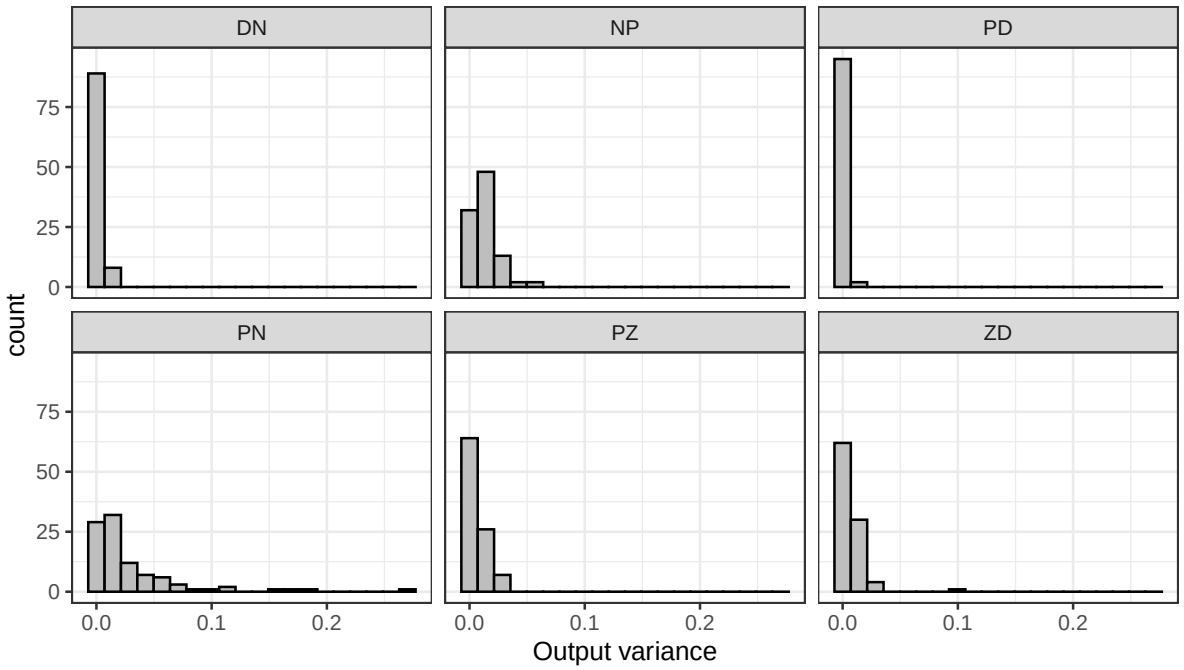


Figure 4.9: Distributions of the variance of the output (time point 365 of PON) as six of the intermediate variables are perturbed at different points in intermediate space. One value here corresponds to the spread represented by one boxplot in Figure 4.8.

We have demonstrated our method using OG99NPZD, an ocean carbon cycle simulator, where the intermediate variable emulation structure enabled us to rule out portions of the input space, and also shed light on which inputs and sub-processes were most active. However, the method is not tied to any particular application and we feel that it represents an important development in the field of modelling uncertainty in complex models.

References.

[1] L. S. Bastos and A. O'Hagan, *Diagnostics for Gaussian Process Emulators*, Technometrics, 54 (2009), pp. 425–438.

[2] M. Bayarri, J. Berger, J. Cafeo, G. Garcia-Donato, F. Lui, J. Palomo, R. Parthasarathy, R. Paulo, J. Sacks, and D. Walsh, *Computer model validation with functional output*, The Annals of Statistics, 35 (2007), pp. 1874–1906.

[3] J. O. Berger and L. A. Smith, *On the statistical formalism of uncertainty quantification*, Annual Review of Statistics and Its Application, 6 (2019), pp. 433–460, https://doi.org/10.1146/annurev-statistics-030718-105232, https://doi.org/10.1146/annurev-statistics-030718-105232.

[4] P. S. Craig, M. Goldstein, A. H. Seheult, and J. A. Smith, *Pressure matching for hydrocarbon reservoirs: a case study in the use of Bayes linear strategies for large computer experiments*, Case Studies in Bayesian Statistics, 3 (1997), pp. 37–93.

[5] J. Cumming and M. Goldstein, *Bayes linear uncertainty analysis for oil reservoirs based on multiscale computer experiments*, in The Oxford handbook of applied Bayesian analysis, A. O'Hagan and M. West, eds., Oxford University Press, 2010, pp. 241–270.

[6] J. Cumming and D. Wooff, *Dimension reduction via principal variables*, Computational Statistics and Data Analysis, 52 (2007), pp. 550 – 565, https://doi.org/10.1016/j.csda.2007.02.012.

[7] D. Drignei and M. C. Drignei, *Uncertainty analysis for functional computer output: A simula-*

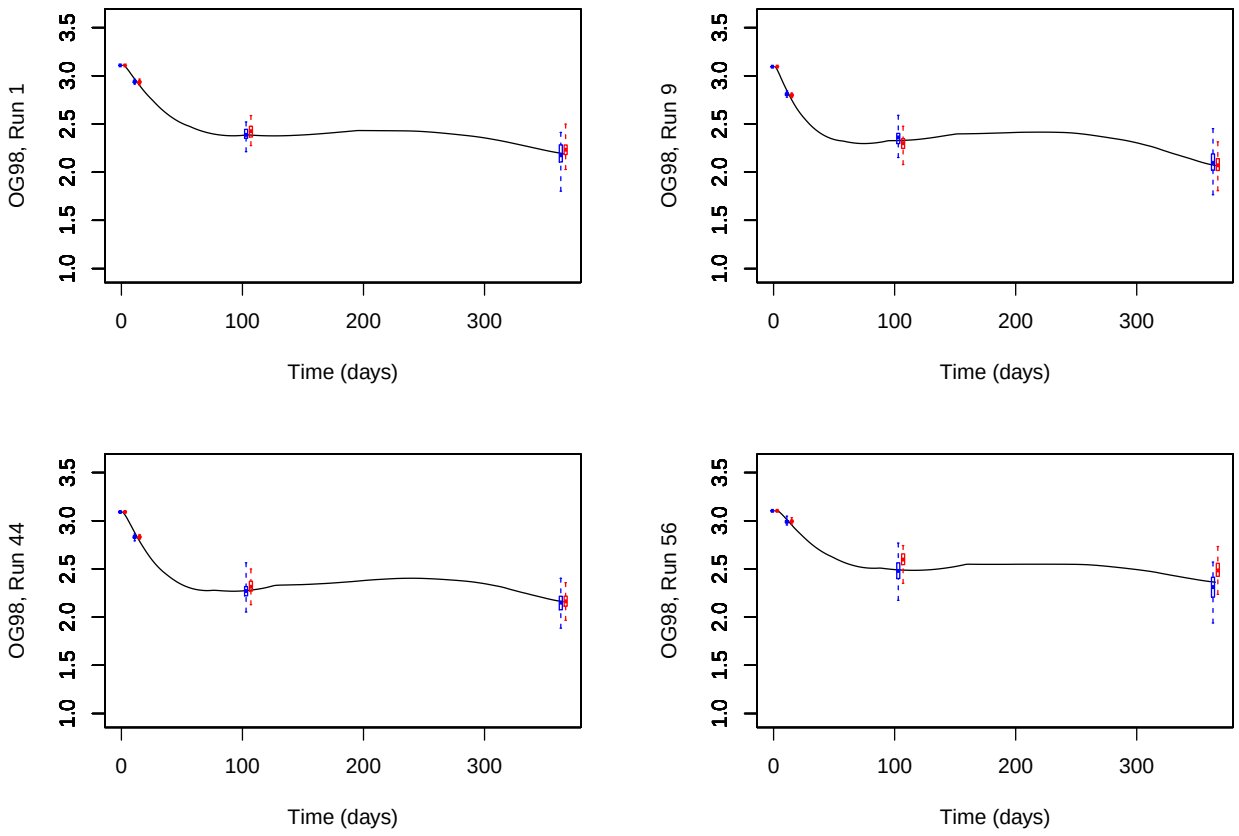


Figure 4.10: Intermediate variable and standard emulators for some new input points. The line shows the output time series of PON, the blue box-plots (left in each pair) summarise 100 samples made using the two-stage method with the intermediate variable emulator at each principal variable time point, the red box-plots (right) summarise 100 draws from a standard emulator.

tion study in inverse problems, Communications in Statistics - Simulation and Computation, 45 (2016), pp. 3281–3293, https://doi.org/10.1080/03610918.2014.932803.

[8] M. Fasham, H. Ducklow, and S. McKelvie, *A nitrogen-based model of plankton dynamics in the oceanic mixed layer*, Journal of Marine Research, 48 (1990), pp. 591–639.

[9] K. Fennel, M. Losch, J. SchrÃűter, and M. Wenzel, *Testing a marine ecosystem model: sensitivity analysis and parameter optimization*, Journal of Marine Systems, 28 (2001), pp. 45 – 63, https://doi.org/https://doi.org/10.1016/S0924-7963(00)00083-X, http://www.sciencedirect.com/science/article/pii/S092479630000083X.

[10] D. Francom, B. Sanso, A. Kupresanin, and G. Johannesson, *Sensitivity analysis and emulation for functional data using Bayesian adaptive splines*, Statistica Sinica, 28 (2018), pp. 791–816, https://doi.org/{10.5705/ss.202016.0130}.

[11] M. Goldstein and N. Huntley, *Bayes linear emulation, history matching and forecasting for complex computer simulators*, Handbook of Uncertainty Quantification, Springer, Cham, Switzerland, (2017).

[12] M. C. Kennedy and A. O'Hagan, *Bayesian Calibration of computer models*, Journal Of The Royal

Statistical Society Series B, 63 (2001), pp. 425–464.
[13] G. Lacroix and M. Grégoire, *Revisited ecosystem model (modecogel) of the ligurian sea:seasonal and interannual variability due to atmospheric forcing*, Journal of Marine Systems, 37 (2002), pp. 229–258.
[14] L. Le Gratiet, C. Cannamela, and B. Iooss, *A bayesian approach for global sensitivity analysis of (multifidelity) computer codes*, SIAM/ASA Journal on Uncertainty Quantification, 2 (2014), pp. 336–363, https://doi.org/10.1137/130926869.
[15] G. P. McCabe, *Principal variables*, Technometrics, 26 (1984), pp. 137–144.
[16] J. E. Oakley and A. O'Hagan, *Probabilistic sensitivity analysis of complex models: A Bayesian approach*, Journal Of The Royal Statistical Society Series B, 66 (2004), pp. 751–769.
[17] A. O'Hagan, *Bayesian analysis of computer code outputs: A tutorial*, Reliability Engineering and System Safety, 91 (2006), pp. 1290–1300.
[18] A. Oschlies and V. Garçon, *An eddy-permitting coupled physical-biological model of the North Atlantic. 1. Sensitivity to advection numerics and mixed layer physics*, Global Biogeochemical cycles, 13 (1999), pp. 135–160.
[19] R. H. Oughton and P. S. Craig, *Hierarchical Emulation: A Method for Modeling and Comparing Nested Simulators*, SIAM-ASA JOURNAL ON UNCERTAINTY QUANTIFICATION, 4 (2016), pp. 495–519, https://doi.org/{10.1137/15M1007914}.
[20] R. H. Oxlade (now Oughton), *Comparing multiple simulators using Bayesian emulators*, PhD thesis, Department of Mathematical Sciences, University of Durham, 2012.
[21] J. Palmer and I. Totterdell, *Production and export in a global ocean ecosystem model*, Deep-Sea Research, 48 (2001), p. 1169âĂŞ1198.
[22] C. Prieur, L. Viry, E. Blayo, and J.-M. Brankart, *A global sensitivity analysis approach for marine biogeochemical modeling*, Ocean Modelling, 139 (2019), p. 101402, https://doi.org/https://doi.org/10.1016/j.ocemod.2019.101402, http://www.sciencedirect.com/science/article/pii/S1463500318303688.
[23] J. M. Salter and D. Williamson, *A comparison of statistical emulation methodologies for multi-wave calibration of environmental models*, Environmetrics, 27 (2016), pp. 507–523, https://doi.org/10.1002/env.2405.
[24] T. J. Santner, B. J. Williams, and W. Notz, *The design and analysis of computer experiments*, Springer, New York, 2003.
[25] M. Schartau and A. Oschlies, *Simultaneous data-based optimization of a 1d-ecosystem model at three locations in the north atlantic: Part iâĂŤmethod and parameter estimates*, Journal of Marine Research, 61 (2003), pp. 765–793.
[26] M. Schartau, P. Wallhead, J. Hemmings, U. Loeptien, I. Kriest, S. Krishna, B. A. Ward, T. Slawig, and A. Oschlies, *Reviews and syntheses: parameter identification in marine planktonic ecosystem modelling*, BIOGEOSCIENCES, 14 (2017), pp. 1647–1701, https://doi.org/10.5194/bg-14-1647-2017.
[27] B. Silverman, *Some aspects of the spline smoothing approach to non-parametric regression curve fitting*, Journal Of The Royal Statistical Society Series B, 47 (1985), pp. 1–52.
[28] M. Strong and J. E. Oakley, *When is a model good enough? deriving the expected value of model improvement via specifying internal model discrepancies*, SIAM/ASA Journal on Uncertainty Quantification, 2 (2014), pp. 106–125.
[29] M. Strong, J. E. Oakley, and J. Chilcott, *Managing structural uncertainty in health economic decision models: a discrepancy approach*, Journal Of The Royal Statistical Society Series C, 61 (2012), pp. 25–45.
[30] J. VanMilgen, M. Murphy, and L. Berger, *A compartmental model to analyze ruminal digestion*, Journal of Dairy Science, 74 (1991), pp. 2515–2529.
[31] I. Vernon, M. Goldstein, and R. Bower, *Galaxy formation: Bayesian history matching for the observable universe*, Statistical science, (2014), pp. 81–90.

[32] I. Vernon, M. Goldstein, and R. G. Bower, *Galaxy formation : a bayesian uncertainty analysis.*, Bayesian analysis., 05 (2010), pp. 619–670.
[33] I. Vernon, J. Liu, M. Goldstein, J. Rowe, J. Topping, and K. Lindsey, *Bayesian uncertainty analysis for complex systems biology models: emulation, global parameter searches and evaluation of gene functions*, BMC SYSTEMS BIOLOGY, 12 (2018), https://doi.org/10.1186/s12918-017-0484-3.
[34] H. Watabe, Y. Ikoma, Y. Kimura, M. Naganawa, and M. Shidahara, *PET kinetic analysis - compartmental model*, Annals of Nuclear Medicine, 20 (2006), pp. 583 – 588.
[35] D. B. Williamson, A. T. Blaker, and B. Sinha, *Tuning without over-tuning: parametric uncertainty quantification for the nemo ocean model*, Geoscientific Model Development, 10 (2017), pp. 1789–1816.
[36] J. Zhang, J. Lou, Z. Ma, and J. Wu, *A compartmental model for the analysis of sars transmission patterns and outbreak control measures in china*, Applied Mathematics and Computation, 162 (2005), pp. 909–924.