# Setting up your simulator

Jonathan Rougier*

School of Mathematics

University of Bristol

Compiled October 17, 2015, from `SUYS3.tex`

**Abstract**

The parameters of a computer simulator are often poorly defined, and their ranges for a particular application can be mysterious. But the negative consequences of getting these ranges wrong, either too small or too large, endure through all subsequent uses. So 'setting-up', by which I mean the process of adjusting the individual ranges in the light of a few carefully-chosen observations, is a crucial first step. Happily there is a relatively straightforward process for setting-up, based on the notion of 'implausibility', and making use of simple calculations and visualisations. This process works much better if the analyst is able to proceed sequentially, through several waves of runs. The paper also considers the extension of this process to different types of simulator: simulators with large fields of parameters, stochastic simulators, and expensive-to-run simulators.

## 1 Introduction

I doubt that the benefits of representing a complex system as a computer model need to be reiterated for the readers of this Journal. So I start by

*School of Mathematics, University Walk, Britol BS8 1TW, UK. Email `j.c.rougier@bristol.ac.uk`. Webpage `http://www.maths.bris.ac.uk/~mazjcr/`.

supposing that you have taken the trouble to construct such a model, and I describe some ways in which such a model might be brought into useful service. My general affiliation for this topic is to the 'Durham school' of history matching (section 5), but much of the material is original. So this paper is partly a review, but mainly a reflection on my own practice, and mistakes. I hope it is helpful in fixing concepts and terms, even if you, the reader, think that some of the suggestions are a bit 'out there'.

Here is an outline of the sections that follow. Section 2 provides some basic concepts and terms for computer models, which I prefer to call 'simulators', and provides some examples. The focus in the following sections is to set lower and upper limits for each of the simulator's 'parameters', which I refer to as 'setting-up'. Section 3 describes the objective when setting-up a simulator, contrasting my approach with a more probabilistic approach, and rejecting the latter at this early stage of the process of developing the simulator. Similarly, I explain why setting-up has to come before screening and global sensitivity analysis. Section 4 gives a set of sanity checks to be performed at the start of the process of setting-up.

Section 5 gives a formal treatment of the calculus of 'implausibility' in terms of confidence sets, and explains how the implausibilities of an ensemble of runs can be used to constrain the ranges of individual parameters. Sections 6 and 7 are the heart of the paper, describing an iterative and visual approach to setting-up, which proceeds in waves of runs. The two critical issues are discussed together in section 6: how many runs to do in the next wave and which observations to use in order to compute implausibility. Visualisation is by Parallel Coordinates Plot, discussed in section 7.

The final four sections cover other issues that frequently arise in applications. First, the crucial issue of how to handle a field of parameter values, such as the Manning's $n$ values along the reaches of a channel: section 8. Second, the need at some stage, usually late in the experiment, to generate a large number of acceptable parameter sets: 'last gasp' design in section 9. Third, stochastic simulators, what they represent and how implausibility can be adapted to them: section 10. Finally, the use of emulators: section 11. The paper ends with a Glossary, and the first occurence of each term in the

Glossary is given in italics.

The supplementary information contains an R worksheet (R Core Team, 2013) which demonstrates the workflow of setting-up, contains useful functions, and produces the figures in the paper.

## 2 Simulators

I find it helpful to reserve the word *simulator* for the computer model: 'model' is already heavily overloaded (Rougier *et al.*, 2013a). 'Model' might conveniently be reserved for the representation of the system on which the simulator is based. In this usage, Beven (2009, sec. 1.4) distinguishes between the 'perceptual' model, the 'formal' model, and the 'procedural' model (the algorithm that the simulator executes). These first two models are also termed the 'conceptual' model and the 'mathematical' model. None of this is germane in what follows.

Figure 1 illustrates what might be termd a 'kitchen sink' simulator, which has all of the features that may be found in a simulator. I introduce some common terms. The *inputs* are the forcing, boundary, parameters, and initial values. One *run* of the simulator maps the inputs to the *outputs*. In a *stochastic* simulator, two runs at the same inputs do not necessarily produce the same outputs; otherwise the simulator is *deterministic*. A well-designed stochastic simulator will make the random number seed available as an additional input, in order that the user can control this source of variation (e.g. for reproducibility and checking); for simplicity, I have not shown this in Figure 1. Stochastic simulators are discussed in section 10.

A hydrological simulator of a floodplain has all of the inputs of this kitchen sink simulator. The boundary is the topography of the flood plain, tiled into pixels. The parameters are Manning's $n$ values (roughnesses), one for each pixel. The initial values are the depths of the surface water for each pixel. The forcing $f = (f_1, f_2, \dots)$ is a sequence of net fluxes (i.e. change in depth), where each $f_t$ is a vector of fluxes at time $t$, one for each pixel. The outputs $x = (x_1, x_2, \dots)$ are a sequence of water depths, where each $x_t$ is a vector of depths at time $t$, one for each pixel—this would be the entire 'state vector' for

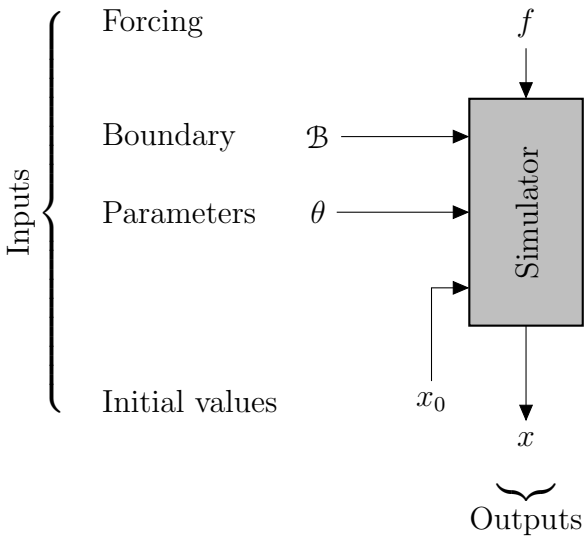


Figure 1: The 'kitchen sink' simulator, which has all of the features that may be found in a simulator. Each of these symbols may represent a large collection of values.

the simulator. See Hunter *et al.* (2007) and Neal *et al.* (2009) for illustrations of this type of simulator. In large applications each pixel might be classified as one of a small number of types, with each type given a common Manning's $n$; this greatly reduces the number of parameters.

The floodplain simulator is an example of a dynamical simulator. For such simulators, the need for initial value $x_0$ (which can be high-dimensional) can sometimes be finessed by 'spinning up' the simulator, which involves starting the simulation well before time $t = 0$, say at time $t = -s$, at a simplistic initial value $x_{-s}$, and then allowing realistic forcing $(f_{-s+1}, \ldots, f_0)$ to erase the memory of $x_{-s}$. For example, the floodplain simulator could be spun-up over a quiet period in which the forcing was the net inflow and outflow of the river channels at the points where they enter and exit the floodplain, and zero elsewhere. Spinning-up is common practice with climate simulators, in which a preindustrial year such as 1750 is used again and again to bring the state vector into something like a physically plausible configuration (Rougier and Goldstein, 2014). Note that spin-up needs to be performed for each distinct

value of the parameters, which is a major computational cost in perturbed parameter experiments on climate simulators (see, e.g., Murphy *et al.*, 2011).

In a simulator, all of the inputs tend to be adjustable, possible excluding the boundary for really large simulators, where it might be hardwired in, for speed. The same core code can be used to represent many different applications of the same system. For example: different topographies, different flow patterns, different flood defences, or different control regimes for flood defences. I would tend to reserve 'boundary' for static features of the system and 'forcing' for the more variable features of the system; e.g., those that could vary in time. This is not an important distinction, though.

Parameters tend to correspond to non-measurable aspects of the system, which is how they are distinguished from forcing or boundary values, and they tend to be static in dynamical simulators. Often, parameters are 'lumped' quantities standing in for processes which are not well-understood, or which operate on a scale below that of the simulator's solver. Our beliefs about parameters tend to be much vaguer than about the other inputs, which correspond to measurable aspects of the system.

As another example, from the other end of the scale to the floodplain simulator, consider a simulator for the steady-state velocity profile of a snow avalanche, as used in Rougier and Kern (2010). The boundary is the slope angle, the forcing is the snow depth and density, and the four parameters control the snow rheology and the interaction between the snow and the underlying surface. The output is steady-state velocity as a function of slope-normal height. This simulator can easily be implemented in a spreadsheet.

We still have to connect the simulator to the system it represents. Figure 2 adds observations to the kitchen sink simulator. *Observables* are specified functions of the simulator outputs, corresponding to operationally-defined system quantities, and *observations* are the measured values of these quantities. This distinction is practically useful, as will be demonstrated below.

The observables represent 'points of contact' between the simulator and the underlying system. The correspondence between observables and system quantities is not exact, but it must be quantifiable. Is this reasonable? I suggest it is—a simulator with no points of contact with the system is not

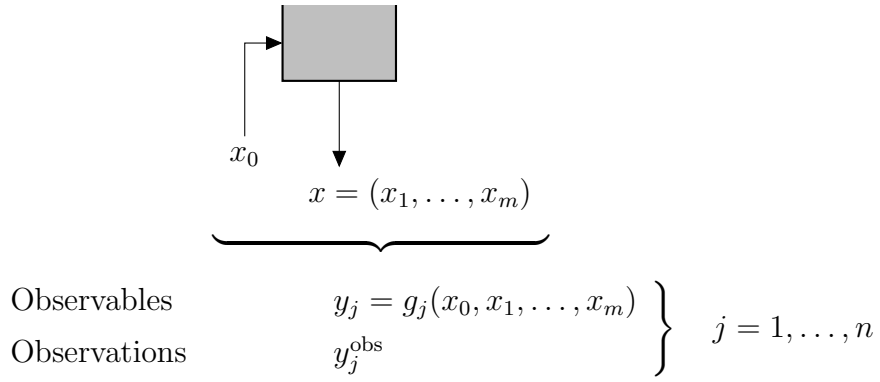


Figure 2: The bottom of the 'kitchen sink' simulator, with added observables and observations. When focusing on the role of the parameters, I write $y_j(\theta) := g_j\big(x_0, x_1(\theta), \ldots, x_m(\theta)\big)$.

capable of providing useful quantitative assessments of system behaviour. If you think that your simulator *can* provide such an assessment, then there must be at least one observable for which you can say, e.g.,

> I believe that there is a parameter value $\theta^*$ for which the simulator observable $y_j(\theta^*)$ is likely to be within $\pm 1.5\,\text{m/s}$ of the true value in the system, given knowledge of the other inputs.

The value $\theta^*$ is not known to you. Its purpose is to represent the idea that the tolerance $\pm 1.5\,\text{m/s}$ is the best you think your simulator can do for this observable: effectively, you do not think you can reduce this value through more careful tuning of the parameters. If you are not able to provide at least one such tolerance, then other people ought to be suspicious about whether your simulator provides any useful quantification of system behaviour. They might think that you wanted to have your cake and eat it too—all the fun of building a simulator, and none of the responsibility of thinking about how it relates to the system.

The use of known functions includes simple observables such as $y_j = x_{ti}$, the $i$th output at time $t$, but in many applications the observables are summaries such as averages. In Zammit-Mangion *et al.* (2014), for example,

there are single pixel measurements of elevations from GPS, 'small footprint' measurements from LIDAR, which give elevation averages over a neighbourhood of pixels, and 'large footprint' measurements from the GRACE satellite, which give mass averages over hundreds of pixels.

But we can also be much more creative in our observables, using highly non-linear functions if that better suits our beliefs. Dynamical simulators provide a good example. The simulator's time-series of outputs might well have a complex relationship with the system's time-series. For example, it might be out-of-phase in a way that depends on the values of the parameters. In this case, observables which are not affected by mis-phasing would be useful. For example, the peak value of the time-series in a window, or perhaps the elapsed time between successive peaks, or perhaps the shape of the peak (e.g. the exponential coefficient of the run-down). Ideally, the observable has the property that its tolerance is relatively invariant to the value of the observable. Sometimes this can be achieved by taking logs, if tolerances are more naturally expressed in terms of percentages.

For the floodplain simulator, one set of observables is the maximum water height at a set of locations following a flood (implemented in the simulator in terms of precipitation and flow forcing), which can be collected from marks on structures after the water has subsided (Neal *et al.*, 2009). This would be an appropriate set of observables for hazard assessment if the main loss from flooding was derived from the footprint of inundation, rather than the speed with which the water level rose and fell. For the avalanche simulator, optical sensors installed in an experimental chute can be used to infer speed at a fixed set of slope-normal heights, in an experiment where the slope and the snow depth and density are all known (Tiefenbacher and Kern, 2004). Much more ambitious experimental sites have also been developed (see, e.g., Maggioni *et al.*, 2013).

I have focused on the observable outputs but the forcing and the boundary are also part of the system. These values must be specified before the simulator is run, and so there is a presumption that they are either measurable or specified in some way (e.g. future climate scenarios), which I have adopt here. But the next section will start with a more general description

of simulator experiments, in which all inputs and outputs are potentially uncertain, and only some are observed, possibly with error.

## 3 Objectives

Most simulator experiments can be described in the same way: some inputs are specified, some inputs and/or observables are measured, possibly with error, and some of the remaining inputs/outputs are to be reconstructed. The language of probability provides a simple picture of such an experiment. A stochastic simulator provides a conditional distribution for the outputs given the inputs (in this context, a deterministic simulator is just a special case). A joint distribution over inputs and outputs is therefore induced by providing a marginal distribution for the inputs. This joint distribution can be conditioned on the measured values of the inputs and on the observations using Monte Carlo simulation methods (see, e.g., Robert and Casella, 2004), and the result can be marginalised to provide a probabilistic reconstruction of the inputs/outputs of interest. Rougier (2007) provides a detailed treatment of this process for a climate simulator.

This probabilistic approach presupposes a well-specified distribution for the inputs, notably the parameters. This cannot be assumed, in the early stages of constructing a new simulator, or at the point of using existing core code with a new set of forcing and boundary values. A similar problem besets less-probabilistic approaches such as GLUE (see, e.g., Beven, 2009, sec. 4.5). Therefore I will not adopt such approaches in this paper. Instead, I will focus on the task of constraining the inputs according to observed values of (some of) the inputs and outputs. For simplicity I will focus on constraining the parameters, treating the forcing and boundary values as known. For the same reason I will initally assume that the simulator is deterministic (relaxed in section 10), and that it can be run many times (relaxed in section 11). There are some fields, like hydrology, where uncertainty in the forcing is an unavoidable and important feature. To some extent this can be addressed by a careful choice of observables (see above, at the end of section 2). Another option is to add a stochastic component to the forcing (see section 10).

What would a well-set-up simulator look like? The standard situation, representing the vast majority of experiments, would be a piece of code (or just an algorithm), and ranges for each of the parameters, suitable for a specified set of forcing, boundary, and initial values. Let $\mathcal{R}_k := (\ell_k, u_k)$ represent the range for the $k$th parameter, and

$$\mathcal{R} := \mathcal{R}_1 \times \cdots \times \mathcal{R}_p$$

represent the parameter space: a $p$-dimensional box, for a $p$-vector of parameters. Why a box? A box is the simplest way to represent a $p$-dimensional space, being parsimonious to conceive and represent, easy to fill, and easy to adjust. But of course a box might not be the 'right' shape for the parameters. Thus the convention must be that the exterior of the box defines a 'ruled out' region: it is necessary for the parameter values to be inside the box, but not sufficient. *Setting-up a simulator is firstly about creating a box $\mathcal{R}$ with this property; secondly, it would be better if the box were as small as possible.* This is what this paper is about.

It is important to distinguish setting-up from screening and global sensitivity analysis. If the ranges of the parameters are already well-constrained, then screening can be used to identify the active parameters (see, e.g., Fitzgerald *et al.*, 2011; Pianosi *et al.*, 2015). But if the ranges are dubious, then screening may do no more than identify parameters whose ranges are too large, and fail to identify parameters whose ranges are too small. So the first task is to get the ranges about right. The practical difference is that screening can proceed without any observations, but setting-up cannot. But it is a delusion that you can therefore do screening before collecting observations, unless you already have strong beliefs about $\mathcal{R}$.

In the rest of this section I consider some trickier cases. Sometimes parameters take discrete values (see, e.g., Table 1 in Rougier *et al.*, 2009b) and so their range is not convex; some are levels of a factor and not directly interpretable on a numerical scale. The methods I outline in this paper can be extended to these parameters, but for simplicity I am not going to consider them here.

Sometimes there are constraints on the parameters. Simple ordering constraints like $\theta_1 \leq \cdots \leq \theta_k$ can be elegantly handled as follows. First, ignore the constraint when generating $\theta_1, \ldots, \theta_k$ values (which presumably all have the same range). Then reorder these values from the smallest to the largest. See Cox and Hinkley (1974, Appendix 2) for the statistical justification for this technique, and Rougier *et al.* (2015) for an application. With more complicated constraints, one possibility is for the simulator to return `NA` for values of $\theta$ not satisfying the constraints. While this is crude, it is entirely compatible with the approach I describe below.

Some runs may not complete. It is very important to find this out before launching a large batch of runs and going away for the weekend. Edwards *et al.* (2011) started with a large experiment which took much less time than we anticipated, because many of the runs failed during the spin-up. This was not a good outcome. Non-completers should be treated suspiciously, as possible evidence of a coding error. This is a subtle issue. The numerical solvers in climate simulators are tuned to particular values of the parameters, and do not take well to large parameter perturbations (and definitely not to several large perturbations in the same run). Does a failure of the solver indicate a ruled-out value for the parameters? In the case of Edwards *et al.* we inspected the outputs and decided that the solver was failing due to a non-physical state vector, and so we classified the parameter values of non-completers as *a fortiori* ruled-out. Such non-completers are not removed from the experiment, but their outputs are recorded as `NA`; see section 9.

## 4 Sanity check

I inserted this section after reflecting on my own failings (as a statistician).

The trick with setting-up is not to waste too much time, or too many CPU cycles. If you are like me, you will have to resist the urge to plunge in and do a really cool experiment. With every simulator there are important sanity checks to do, even in cases where the core code has been used hundreds of times. In my experience, the problem is not usually with the core code, but with the 'wrapper' that connects the core code with the inputs, extracts

the outputs, and computes the observables. For example, in a large and complex simulator, like a climate simulator, it is easy to miss that a non-default initialisation file was not loaded. Defensive coding can help, plus very detailed log files produced during the run and stored alongside the output files. This is a heartfelt recommendation because tracking down a wrapper error in raw output is a tedious and time-consuming task.

Here is a minimal set of sanity checks for a simulator. They apply across the entire set of parameters, but where some of the parameters are a field of values, it makes more sense to apply them to a subset after reparameterising; see section 8. The starting point is a tentative assessment of $\mathcal{R}$, the parameter box, plus a short list of important observables, with their observations. 'Important' depends on context, but they should be sensitive to key parameters, and similar to the kinds of observables that will be used in later stages of the analysis. The analysis in section 5 suggests that it is better if these observables are qualitatively different one from another.

1. Do a run at the midpoint of $\mathcal{R}$.

2. Do $2p$ axial runs, taking each parameter down to its lowest value and up to its highest value, with the other parameters at their central values.

   Reflect on the values of the observables from these axial runs.

3. Lo-hi runs: for each of a set of important observables do the following two runs:

   (a) Set each parameter at the value (low, midpoint, or high) which gives the lowest value of the observable, based on the midpoint run and that parameter's two axial runs.

   (b) The same, but for the highest value of the observable.

   You may not need to do one or both of these runs, because there may be duplication. Check that the observable value from the first run is lower than the observable value from the second run, and that the observation lies inside this interval.

These sanity checks are neither necessary or sufficient, because non-linearities and interactions in the simulator could cause some weird effects. But it would be good to know about this as soon as possible. Where the observations are not inside the interval, I would consider extending one or more of the ranges, rather than assuming that non-linearities and interactions will come to my rescue—I would be concerned about my ability to set up a simulator where this assumption was required. Mind you, some of the non-linearities and interactions may fade in importance as the size of $\mathcal{R}$ is reduced, so setting-up might get easier if the first few waves (see section 6) have a strong effect on some of the parameter ranges.

It is helpful to have a running example, to illustrate this design and the analysis in sections 6, 7, and 9. So consider a simulator with five parameters, each with $\mathcal{R}_k \leftarrow (0, 1)$, and for which the first observable has the form

$$y_1(\theta) \leftarrow 3\theta_1^{0.5} - \{1 + 4(\theta_1 - 0.5)^2\}\theta_2 + \theta_3(1 - \theta_4)\theta_5. \tag{1}$$

The observation is $y_1^{\text{obs}} \leftarrow 1.84$. The sanity check runs are shown in Table 1. The sanity checks all pass. There is evidence of non-linearity in $\theta_1$ from the axial runs. There is evidence of interactions from the lo-hi runs, inferred as follows. The (local) main effect of $\theta_1$ is $2.12 - (-0.88) \approx 3$, and the main effect of $\theta_2$ is $2.25 - 1.25 \approx 1$. The main effects of the other three parameters are smaller, $1.87 - 1.62 \approx 0.3$, say a total of $\approx 1$ between them. The total effect from the lo-hi runs is $4.00 - (-2.00) \approx 6$, which is 20% larger than the sum of the main effects, $\approx 5$. The suspicion would be a $\theta_1 \cdot \theta_2$ interaction, given that these are the two parameters with large main effects.

At the end of the sanity check, you should be happy that your $\mathcal{R}$, possibly after adjustment and some re-running, is not too small. It is likely to be much too large. Doing runs that span an $\mathcal{R}$ which is much too large is extremely wasteful, which strongly suggests proceeding iteratively. This is discussed in sections 6 and 7.

Table 1: Sanity check runs for observable $y_1$. The first run is a midpoint, the next ten runs are axial runs for the five parameters, and the final two runs are the lo-hi runs for $y_1$.

| $\theta_1$ | $\theta_2$ | $\theta_3$ | $\theta_4$ | $\theta_5$ | $y_1$ |
|---|---|---|---|---|---|
| 0.5 | 0.5 | 0.5 | 0.5 | 0.5 | 1.75 |
| 0 | 0.5 | 0.5 | 0.5 | 0.5 | −0.88 |
| 1 | 0.5 | 0.5 | 0.5 | 0.5 | 2.12 |
| 0.5 | 0 | 0.5 | 0.5 | 0.5 | 2.25 |
| 0.5 | 1 | 0.5 | 0.5 | 0.5 | 1.25 |
| 0.5 | 0.5 | 0 | 0.5 | 0.5 | 1.62 |
| 0.5 | 0.5 | 1 | 0.5 | 0.5 | 1.87 |
| 0.5 | 0.5 | 0.5 | 0 | 0.5 | 1.87 |
| 0.5 | 0.5 | 0.5 | 1 | 0.5 | 1.62 |
| 0.5 | 0.5 | 0.5 | 0.5 | 0 | 1.62 |
| 0.5 | 0.5 | 0.5 | 0.5 | 1 | 1.87 |
| 0 | 1 | 0 | 1 | 0 | −2.00 |
| 1 | 0 | 1 | 0 | 1 | 4.00 |

## 5 Implausibility

A parameter value $\theta \in \mathcal{R}$ is *implausible* for you exactly when the value of at least one observable is further from its corresponding observation than is consistent with your beliefs about the relationship between the simulator and the system. Your beliefs are themselves subject to revision through the process of screening. For example, if you find that the whole of $\mathcal{R}$ is implausible, as far as you can tell, then perhaps the simulator is not as good as you think it is (good advice in this situation is provided by Beven, 2009, sec. 4.5.9). Implausibility was introduced by Craig *et al.* (1996, 1997), and discussed more recently in Vernon *et al.* (2010). A related concept is the more qualitative behavioural/non-behavioural distinction, separating parameter values which give the right sort of behaviour from those that do not (see, e.g., Beven, 2009, sec. 3.5.4, for a discussion and references). Implausibility can be seen as a generalisation of this approach, providing a numerical scale from 'behavioural' to 'non-behavioural' which can be manipulated (combined, projected) in accordance with statistical rules.

Suppose that there is such a thing as a 'best' value of the parameter, denoted $\theta^*$. I will not explore the meaning of 'best' in this paper: see Goldstein and Rougier (2004, 2009) for a discussion. Let's just agree that this is a helpful concept, without getting too bogged down. As already stated in section 2, the existence of $\theta^*$ is asserted, but its value is unknown.

Consider the $j$th observable. The tolerable distance between the observation $y_j^{\text{obs}}$ and the 'best' simulator value $y_j(\theta^*)$ obviously includes a contribution from *measurement error*. It also includes a contribution from *representation error*, which is the incommensurability between a simulator observable and a system observation (e.g. between a pixel and a point). Plus a contribution from *structural error*, which represents the limitations of the simulator (see, e.g., Rougier and Beven, 2013). For simplicity I will assume that structural error is insensitive to the value of $\theta^*$; this has already been discussed in section 2, concerning good choices of observables, and is discussed further at the end of this section.

Taking our cue from statistical errors, we can quantify each of these con-

tributions to the difference between $y_j^{\text{obs}}$ and $y_j(\theta^*)$ as standard deviations and combine them by adding their squared values, to get a single scale for tolerable distance, denoted $\sigma_j$. Implicitly we are treating each of these errors for observable $j$ as independent, in simply adding their variances; this seems uncontroversial. In this sum of three variances, one large term might dominate—typically this might be structural error, for a simulator which represents a complex system. In practice, though, many people seem to use just measurement error (e.g. in scaling RMSE), which sets the bar for the parameters far too high. It is not sensible to judge the parameters of a simulator that is known to have structural limitations by the standards of one that does not.

As stated in section 2, if you cannot find one observable for which you can quantify a tolerance, then other people ought to be suspicious about whether your simulator provides any useful quantification of system behaviour. If you can find one such observable, then you ought to be able to find a set of them with only a little more effort. I reiterate the point made above, that you may end up revising your values for the tolerances in the light of things you learn about the simulator when you run it—they are not set in stone.

This standard deviation $\sigma_j$ can be used in a unitless measure of implausibility for parameter $\theta$ based on observation $j$,

$$I_j(\theta) := \frac{\left|y_j^{\text{obs}} - y_j(\theta)\right|}{\sigma_j} \quad \text{where} \quad \sigma_j := \sqrt{\sigma^2_{\text{meas},j} + \sigma^2_{\text{repr},j} + \sigma^2_{\text{str},j}}\,; \qquad (2)$$

if $y_j(\theta) = \texttt{NA}$, then $I_j(\theta) \leftarrow \infty$. General results about standardised deviations from the expectation of a random quantity can be used to give upper bounds for the probabilities of large implausibiliites.

Take Chebyshev's inequality. This states that an implausibility $I_j(\theta) \geq 3$ has a probability of not more than $1/9 \approx 0.11$ of occurring under the null hypothesis $H_0 : \theta^* = \theta$. A tighter result is possible if your beliefs about the difference between $y_j^{\text{obs}}$ and $y_j(\theta^*)$ can be represented by an absolutely continuous and unimodal distribution (e.g. a Normal distribution). In this case an implausibility of at least 3 has a probability of not more than 0.05 of occuring under the null hypothesis (the '$3\sigma$ rule', see Pukelsheim, 1994, and

below). Stronger still, if your beliefs can be represented by an actual Normal distribution, then an implausibility of at least 3 has a probability of 0.003 under the null hypothesis.

In turn, these probability bounds can be turned into confidence sets for $\theta^*$. The set

$$\mathcal{C}_j := \left\{\theta \in \mathbb{R}^p : I_j(\theta) \leq 3\right\} \tag{3}$$

is a level 89% confidence set for $\theta^*$ according to Chebyshev, a 95% confidence set according to the $3\sigma$ rule, and a 99.7% confidence set according to the Normal distribution. The formal reasoning is given at the end of this section.

The set $\mathcal{C}_j$ uses just one observable. It would be better to use more, but there is a difficulty. There are dependencies between the structural errors of different observables, because simulators tend to be systematically wrong, even when run at the 'best' value of the parameters. In an extreme case, the structural error may be the same in all observables; e.g. the whole set of outputs may be vertically shifted by an unknown amount. The presence of dependencies makes it hard to assess the level of a confidence set created by combining individual confidence sets; hard, but not futile. If the components of error for observable $i$ are effectively independent of those for observable $j$ then

$$\begin{aligned}\Pr\{\theta^* \in \mathcal{C}_i(Y_i) \cap \mathcal{C}_j(Y_j); \theta^*\} &= \Pr\{\theta^* \in \mathcal{C}_i(Y_i); \theta^*\} \cdot \Pr\{\theta^* \in \mathcal{C}_j(Y_j); \theta^*\} \\ &\geq 0.95 \cdot 0.95 \approx 0.90,\end{aligned}$$

say, if each $\mathcal{C}$ is defined using (3) under the $3\sigma$ rule (the notation comes from the explanation at the end of this section). Under a positive dependence, the lower bound would be higher, possibly as high as 0.95.

If we take two independent confidence sets defined using (3) and combine them, then their intersection has level 79% (Chebyshev), 90% ($3\sigma$, as shown above), or 99.4% (Normal). Although it is rather heuristic, if your beliefs lie somewhere between $3\sigma$ and Normal, then a few (say three or four) intersections will push the level down to about 95%. So let $J$ be a subset of a few observables that are qualitatively different one from another, in the sense

that you believe their structural errors are effectively uncorrelated. Starting with (3), since $\theta \in \bigcap_{j\in J} \mathcal{C}_j$ if and only if $I_j(\theta) \leq 3$ for all $j \in J$, the joint implausibility measure and confidence set are

$$I_J(\theta) := \max_{j\in J} I_j(\theta) \quad \text{and} \quad \mathcal{C}_J := \left\{\theta \in \mathbb{R}^p : I_J(\theta) \leq 3\right\}. \tag{4}$$

$\mathcal{C}_J$ defines a roughly 95% confidence set for $\theta^*$ based on observables $J$. In other words, it has a coverage probability of roughly at least 95%, no matter what the value of $\theta^*$ happens to be. I write 'roughly' rather than 'approximately' here and elsewhere to emphasize that this is a *very* crude assessment of the lower bound on coverage.

Confidence sets also have a projection property, which is that if $\mathcal{C}$ is a level 95% confidence set for $\theta^*$, then $g(\mathcal{C})$ is a level 95% confidence set for $g(\theta^*)$. Because we are interested in constraining the box $\mathcal{R}$, our primary interest is in one-dimensional projections (although the following argument also works for higher dimensions). Let $g(\theta) \leftarrow \theta_k$, so that $g(\mathcal{C})$ is the set of $\theta_k$ values for every $\theta \in \mathcal{C}$. It follows straightforwardly that the projected implausibility measure and projected confidence set are

$$I_J(\theta_k) := \min_{\theta_{-k}} I_J(\theta_k, \theta_{-k}) \quad \text{and} \quad \mathcal{C}_{Jk} := \left\{\theta_k \in \mathbb{R} : I_J(\theta_k) \leq 3\right\} \tag{5}$$

where $\theta_{-k}$ denotes all parameters bar $\theta_k$. Thus $\mathcal{C}_{Jk}$ is a roughly 95% confidence set (often an interval) for $\theta_k^*$, based on the observables $J$.

This definition of implausibility, and the aggregration and projection rules given in (4) and (5), were originally discussed in Craig *et al.* (1996, 1997), under the name 'history matching'. Their justification of the threshold of 3 was in terms of the $3\sigma$ rule, and their justification of the rules for aggregation and projection was intuition; other intuitive rules have also been proposed, see Vernon *et al.* (2010). As far as I am aware, this paper offers the first interpretation of implausibility in terms of confidence sets, and explains the conditions under which $\mathcal{C}_J$ in (3) can be interpreted as a roughly 95% confidence set for $\theta^*$ based on the observables $J$, and $\mathcal{C}_{Jk}$ as a roughly 95% confidence set for $\theta_k^*$.

Vernon *et al.* (2010) describe the values of $\theta$ in $\mathcal{C}_J$ as 'Not Ruled Out Yet' (NROY, or 'enroy'). This term reflects the fact that adding another observable to $J$ can never increase the size of the confidence set for $\theta$ or $\theta_k$, according to (4) and (5). Thus confidence sets have a natural affinity with the way that the exterior of the box $\mathcal{R}$ defines a ruled-out region. Ideally we want the smallest box which contains $\mathcal{C}_J$, and this will be the box

$$\mathcal{R}_J := \mathcal{C}_{J1} \times \cdots \times \mathcal{C}_{Jp}, \tag{6}$$

for which we automatically have $\mathcal{R}_J \supset \mathcal{C}_J$. The advantage of this approach to setting the ranges is that $\mathcal{R}_J$ also has the property of being a roughly 95% confidence set for $\theta^*$, recollecting from its definition that a level $\beta$ confidence set has coverage of *at least* $\beta$ for all $\theta^*$.

An attraction of the confidence set approach to implausibility, which is not available for the just-intuitive approach, is that it gives more guidance about the implausibility threshold, as a function of the number of observables in $J$. As Pukelsheim (1994) shows, under the $3\sigma$ conditions the general rule for being $r$ from the expectation is

$$\Pr\{|X - \mu| \geq r\} \leq (4/9)(\sigma^2/r^2),$$

for sufficiently large $r$. So a value $r = 4\sigma$, for example (rather than $3\sigma$), implies an exceedance probability of less than $1/(9 \cdot 4) \approx 0.028$ (rather than $4/9^2 \approx 0.05$), which can be used to contruct roughly 95% confidence sets for a larger number (say, five or six) of clearly different observables. Likewise, $r$ can be adjusted to give a different level of coverage, say 99%.

### Formal reasoning for implausibility and confidence sets

This subsection gives the formal statistical reasoning under which a set such as $\mathcal{C}_j$ defined in (3) can be treated as a 95% 'observed' confidence set. The simulator is used as part of a statistical model for the observables. In this model, there is a 'best' value of the parameters, $\theta^* \in \mathcal{R}$, for which the

statistical model asserts that

$$\mathrm{E}(Y_j;\theta^*) = y_j(\theta^*) \quad \text{and} \quad \mathrm{Var}(Y_j;\theta^*) = \sigma_j^2$$

where $Y_j$ is the observation, thought of at this stage as a random quantity parameterised by $\theta^*$. Then the $3\sigma$ rule (Pukelsheim, 1994) states that

$$\Pr\left\{\frac{|Y_j - y_j(\theta^*)|}{\sigma_j} \leq 3; \theta^*\right\} \geq 0.95 \quad \text{for all } \theta^* \in \mathcal{R} \tag{7}$$

(subject to smoothness conditions on the distribution of $[Y_j;\theta^*]$ which we assume hold). Now define

$$\mathcal{C}_j(v) := \left\{\theta \in \mathbb{R}^p : \frac{|v - y_j(\theta)|}{\sigma_j} \leq 3\right\} \tag{8}$$

where $v$ is any possible value of $Y_j$. Because

$$\frac{|v - y_j(\theta)|}{\sigma_j} \leq 3 \iff \theta \in \mathcal{C}_j(v)$$

for every possible $v$, (7) can be written as

$$\Pr\left\{\theta^* \in \mathcal{C}_j(Y_j); \theta^*\right\} \geq 0.95 \quad \text{for all } \theta^* \in \mathcal{R}$$

and this confirms that $\mathcal{C}_j(\cdot)$ in (8) is a 95% confidence set for $\theta^*$, according to the standard definition (see, e.g., Casella and Berger, 2002, sec. 9.1). Finally, I use the shorthand $\mathcal{C}_j$ for the confidence set associated with the observed value of $Y_j$, from which

$$\mathcal{C}_j := \mathcal{C}_j(y_j^{\text{obs}}) = \left\{\theta \in \mathbb{R}^p : I_j(\theta) \leq 3\right\}$$

according to (8) and the definition of implausibility in (2).

Different assumptions give different values on the righthand side of (7). The weaker assumption that $\sigma_j$ is finite (i.e. without any smoothness) gives the Chebyshev lower bound of $8/9 \approx 0.89$; the much stronger assumption that $Y_j$ is Normal gives the exact value 0.997. There would be no theoretical

difficulty in allowing $\sigma_j$ to vary with $\theta^*$; only the practical difficulty of quantifing this relationship in practice. Both section 10 and section 11 involve generalisations where $\sigma_j$ varies with $\theta^*$, in a way that can be fairly easily quantified.

## 6 Proceeding in waves

After the sanity check runs in section 4 you should have satisfied yourself that your $\mathcal{R}$ is not too small—but if so, it is likely to be too big, maybe much too big. After all, nothing you have done so far has reduced $\mathcal{R}$ from your initial assessment, unless you have got `NA`'s from your simulator in the sanity check runs, and had to shrink one or more of the $\mathcal{R}_k$'s. This section explains why proceeding in *waves* can be beneficial, and how you can make a crude assessment of how many runs to do in each wave.

But before going any further—start running your simulator! Run it on a Sobol sequence over $\mathcal{R}$, and keep it running until you send an interrupt (or exhaust your budget, or the patience of other computer users). Run it on multiple CPUs if you can, so that a few *sticky runs*, which take a long time to complete, do not hold up all of the other runs. Not collecting the sticky runs is a minor 'missing not at random' problem; not collecting many runs at all because the sticky runs held up all of the other runs is a major 'small data' problem. Make sure to trap errors in your simulator and code their outputs as `NA` (i.e. do not discard them, see section 9); again, you do not want them to derail your simulations. I would also code unfinished sticky runs as `NA`, pending further investigation.

What you now have is a *diary management problem (DMP)*. You will need, say, three hours in order to analyse the first wave of runs—three hours might be a bit long to implement the approach in this section and section 7, but it is best to be cautious (take the rest of the afternoon off!). The wave needs to be big enough that you have a realistic expectation of revising $\mathcal{R}$ based on the outputs. So you need to calculate the minimum number of runs you will need in the wave, and then the time for these runs to complete (which you can approximate from the run times you have so far, e.g. from the

Table 2: Notation in section 6.

| | |
|---|---|
| $\theta := (\theta_1, \ldots, \theta_p) \in \mathcal{R}$ | The parameter vector, and parameter space |
| $\theta^*$ | The 'best' value of the parameter vector |
| $y_j(\theta)$, $y_j^{\text{obs}}$ | The $j$th observable and its observation |
| $A_j \subset \{1, \ldots, p\}$ | The active parameters of the $j$th observable |
| $J$ | A set of observables |
| $A_J := \bigcup_{j \in J} A_j$ | The active parameters of the set of observables $J$. |
| $\mathcal{R}_{A_J}$ | The subset of $\mathcal{R}$ corresponding to the parameters in $A_J$ |
| $\mathcal{C}_J \subset \mathcal{R}$ | A confidence set for $\theta^*$ based on the set of observables $J$. |
| $\mathcal{C}_{Jk} \subset \mathcal{R}_k$ | A confidence set for $\theta_k^*$ based on $\mathcal{C}_J$. |

sanity check). Then you can identify the earliest time at which you might send the interrupt, and look in your diary for the first block of three hours after this time. You need to do this calculation even if your simulator is fast; it's just that in this case you may find that you can compress the whole process of running the wave and analysing the results into one session.

You will probably want to iterate this process. You cannot expect to learn about more than one or two parameter ranges in each wave. Typically at least one of the initial $\mathcal{R}_k$'s will be far too large, and runs with $\theta_k$ values spread across the whole of this range will almost all be implausible according to an observable which is sensitive to $\theta_k$. But this presents a good opportunity to shrink $\mathcal{R}_k$. Once shrunk, runs in the new $\mathcal{R}$ can be used to restrict the ranges of other parameters which were previously overshadowed by $\theta_k$.

At the heart of this process is an approach for choosing how many runs to do in the next wave, and which observables to use in the next implausibility assessment. That is the subject of the rest of this section. How to analyse the runs is discussed in section 7.

Consider the $j$th observable. Typically, $y_j(\theta)$ will be more sensitive to some parameters than to others; term these the *active* parameters for $y_j$, de-

noted $\theta_{A_j}$, where $A_j \subset \{1, \ldots, p\}$. Given an observation $y_j^{\text{obs}}$ and a tolerance $\sigma_j$, the set $\mathcal{C}_j$ defined in (3) contains NROY values for $\theta$, and represents, roughly, one constraint on the values of $\theta_{A_j}$. At this point intuition from linear algebra is helpful. If we can find a set of $v$ observables $J$ for which the size of $A_J := \bigcup_{j \in J} A_j$ is no greater than $v$, then $\mathcal{C}_J$ ought to be compact in $\mathcal{R}_{A_J}$ (because there are at least as many constraints as parameters in $A_J$), and at least one of its 1D projections ought to have the property that $\mathcal{C}_{Jk}$ is a strict subset of $\mathcal{R}_k$, indicating that $\mathcal{R}_k$ can be shrunk.

Now this is interesting but a bit speculative. However, you have the means to identify good candidates for $J$ and to decide how many runs to do in order to have a chance of revising $\mathcal{R}_k$ for at least one of the $k \in A_J$. From the complete ensemble of runs, remove all *legacy runs* which have values for $\theta$ outside the current $\mathcal{R}$, and also all runs with `NA` outputs. Now do the following calculation:

1. Standardise the parameter values for the runs so that they lie in the unit cube; i.e. linearly transform each parameter so that $\ell_k$ maps to 0 and $u_k$ maps to 1, where $\mathcal{R}_k = (\ell_k, u_k)$ is the current range for $\theta_k$.

2. For each observable for which you have an observation $y_j^{\text{obs}}$ and a tolerance $\sigma_j$, linearly regress the values of the observable on a constant plus the standardised parameter values, to estimate the regression coefficents, $\beta_{jk}$. These have the same units as $y_j$.

3. For each observable and each parameter, compute $\pi_{jk} := |\beta_{jk}|/\sigma_j$. These are unitless and non-negative.

The $\pi_{jk}$'s show how much moving the value of $\theta_k$ from $\ell_k$ to $u_k$ affects $y_j$ as a proportion of the tolerance $\sigma_j$. A large $\pi_{jk}$ among $\pi_{j1}, \ldots, \pi_{jp}$ indicates that $\theta_k$ is an active parameter for $y_j$. A large $\pi_{jk}$ among the entire set of $\pi_{jk}$'s indicates that observable $j$ is useful for constraining $\mathcal{R}_k$, because it suggests that many values of $\theta_k$ will imply a value of $y_j(\theta)$ which lies outside the set of tolerable values surrounding $y_j^{\text{obs}}$.

If you lay-out the $\pi_{jk}$'s in a table with observables down the rows and parameters along the columns, then you will be able to see at a glance where

the good choices for $J$ are, and which parameters they constrain. This is still an informal process, but you might start with the largest $\pi_{jk}$, find out what the other active parameters for that observable are, find out which other observables also have these active parameters, and so on, until you have compiled a set of observables $J$ with active parameters $A_J$, hoping of course that $|A_J| \leq |J|$.

After applying this procedure you may have several candidate $J$'s. If you are really lucky you will have a $J$ for which $A_J$ will be a singleton, say $\{k\}$, indicating that there are $|J|$ observables which have the same single active parameter $\theta_k$; $|J| = 1$ would be good, but $|J| > 1$ would be better. However, I will not assume this in what follows, but consider the general case. What you would really like to do with your next wave is put some runs into $\mathcal{C}_J$, and plenty of runs outside it. This ought to allow you to shrink $\mathcal{R}_k$ for at least one $k \in A_J$.

Let $n_J$ be the number of runs needed to put at least ten runs in $\mathcal{C}_J$ and at least ten outside it, where 'ten' is just a place-holder for some number which you think appropriate. My simple strategy for finding $n_J$ is to use the linear regressions already computed as a surrogate for the simulator. This would be a bad strategy if we wanted to know the value of the simulator at untried values of the parameters, but all we want to do here is get a feeling for the volume of the parameter space which is occupied by $\mathcal{C}_J$. So let $\tilde{\mathcal{C}}_J$ denote the NROY region based on the linear regressions in standardised parameters, rather than based on the simulator. Then define

$$v_J := \int_{(0,1)^p} \cdots \int \mathbb{1}\{z \in \tilde{\mathcal{C}}_J\} \, \mathrm{d}z_1 \cdots \mathrm{d}z_p \tag{9}$$

which is the proportion of $(0,1)^p$ occupied by $\tilde{\mathcal{C}}_J$, which is a proxy for the proportion of $\mathcal{R}$ occupied by $\mathcal{C}_J$. In practice this integral would be approximated numerically at negligible cost. So a space-filling design of $\lceil 1/v_J \rceil$ runs will put roughly one run in $\tilde{\mathcal{C}}_J$, and, by extension, roughly one run in $\mathcal{C}_J$. I would prefer to put at least ten runs into $\mathcal{C}_J$, not just one, and at least ten

runs outside it, and so I would set

$$n_J \leftarrow 10 \cdot \lceil \max\{1/v_J, 1/(1-v_J)\} \rceil.$$

This calculation can be performed for each candidate $J$. As already discussed, we should prefer those $J$ for which $A_J$ is a singleton (i.e. a single active parameter), and those with $|A_J| \leq |J|$, as both of these will project better into individual $\mathcal{R}_k$'s. I would also prefer $n_J$'s which are smaller, on the grounds that, diary permitting, it is more effective to proceed in many small waves than in a few large ones, where the low-hanging fruit are picked first.

Among the favoured candidates, there may be two (or more) with no common active parameters, i.e. a $J$ and a $J'$ for which $A_J \cap A_{J'} = \emptyset$. If a wave of size $\max\{n_J, n_{J'}\}$ is feasible, then both $J$ and $J'$ can be retained for the analysis in section 7.

Returning to the example of section 4, the only possible choice is $J \leftarrow \{1\}$, simply because this is the only available observable. Let the tolerance for $y_1$ be $\sigma_1 \leftarrow 0.1$. The $\pi_{1k}$ values are then

$$\pi_1^T = (32.1,\ 12.1,\ 4.6,\ 4.6,\ 4.6) \tag{10}$$

indicating that $\theta_1$ is an active parameter for $y_1$, and possibly $\theta_2$ is as well. A scrambled Sobol sequence of $10^4$ points is used to compute $v_J = 0.19$. A cruder calculation based on $\theta_1$ alone gives $v_J \approx 6/\pi_{11} \approx 0.19$, a very similar answer. The value for $v_J$ gives $n_J = 10\lceil 1/0.19 \rceil = 60$. Doing 60 runs in a space-filling design is expected to produce roughly 10 runs that are NROY according to $y_1^{\text{obs}}$.

## 7 Restricting the ranges

Now you are in the situation of having an ensemble of runs, and you anticipate that some of these runs will be NROY according to a set of observables $J$. As before, ignore legacy runs which are outside your current $\mathcal{R}$ and `NA` runs; denote the parameter values of the $n$ runs that remain as $\theta^{(1)}, \ldots, \theta^{(n)}$.

For each of these runs you have computed an implausibility with respect to observables $J$ using (4), denote these as $I_J^{(1)}, \ldots, I_J^{(n)}$. If you have a second set of observables $J'$ (as explained at the end of section 6), then the following approach can be applied separately to each set.

As explained in section 5, around (6), the marginal confidence set $\mathcal{C}_{Jk}$ can be used for the revised value of $\mathcal{R}_k$, for $k = 1, \ldots, p$. The problem is that the operation of maximising the implausibility over all but the $k$th parameter is not available, since it would require multiple further runs of the simulator for each possible value of $\theta_k$. However, this operation can be approximated within the current set of runs because that set is space-filling.

Consider the question of whether $t_k \in \mathcal{C}_{Jk}$. We would like to compute

$$I_J(t_k) = \min_{\theta_{-k}} I_J(t_k, \theta_{-k})$$

to find out whether $I_J(t_k) \leq 3$, according to (5). We can approximate this value with

$$\tilde{I}_J(t_k) := \min \left\{ I_J^{(i)}, \text{ for those } i \text{ for which } \theta_k^{(i)} \in B_\delta(t_k) \right\}, \qquad (11)$$

where $B_\delta(t_k)$ represents an interval of width $2\delta$ centered at $t_k$. In other words, minimise the implausibility over the subset of runs which have values of $\theta_k$ within $\delta$ of $t_k$, where $\delta$ needs to be a small value on the scale of $\mathcal{R}_k$. This approximation works, if it works at all, because the set of runs is space-filling in $\theta$, and hence space-filling in the thick slice of $\mathcal{R}$ around $\theta_k = t_k$ for any $t_k \in \mathcal{R}_k$.

The choice of semi-width $\delta$ in (11) seems to be crucial. Too small and the subset is not large enough to span $\mathcal{R}_{-k}$ effectively; too large and the implausibilities are not relevant to the value $\theta_k = t_k$. Luckily this task can be delegated to the human eye, using a Parallel Coordinates Plot (PCP, see Venables and Ripley, 2010, sec. 11.1). The PCP is described in Figure 3. The eye does the smoothing by seeing systematic patterns in the colours along each of the parameter axes, notably finding the subset of $\mathcal{R}_k$ where there is unlikely to be any low-implausibility values. This subset is likely to be

outside $\mathcal{C}_{Jk}$, and can be (tentatively) excluded from the revised $\mathcal{R}_k$.

Figure 3 includes both the 12 sanity check runs (8 were NROY) and the 60 first wave runs (18 were NROY) from the toy problem. 18 NROY runs is a bit more than the 'roughly ten' we were expecting (we ignored non-linearities and interactions). Focusing on the dark-blue lines, which have $I_J^{(i)} \leq 3$, it looks reasonable to revise both $\mathcal{R}_1$ and $\mathcal{R}_2$. This is a good outcome, because one observable may not be able to constrain two active parameters, as explained in section 6. If $\mathcal{R}_1$ and $\mathcal{R}_2$ are both shrunk to the extremal NROY values, the new ranges are

$$\mathcal{R}_1 \leftarrow (0.27, 1.00) \quad \mathcal{R}_2 \leftarrow (0.05, 0.81)$$

for a new volume for $\mathcal{R}$ which is 56% of the original volume. This is quite an aggressive shrinking, but it is not irreversible. Moreover, the last gasp design (section 9) will tend to push the ranges back out again, if necessary.

There are NROY runs at the edges of $\theta_3$, $\theta_4$, and $\theta_5$. However, the values of $\pi_1$ in (10) suggested that $y_1$ was not likely to be very informative about these three parameters, and so I would leave these ranges unchanged for the second wave.

It is tempting to 'squeeze' $y_1$ further, with some more runs in the shrunken space. However, these runs cannot shrink $\mathcal{R}$ any further, since there are now NROY runs which reach the lower and upper limits of each of the ranges. Further progress will require a new observable.

So, to summarise. From the previous section, choose observables $J$, compute $v_J$ and $n_J$, do a space-filling design of at least $n_J$ runs. In this section, compute the implausibility of each run using $J$, draw the PCP, and, for each $\theta_k$, decide if any of $\mathcal{R}_k$ can be ruled out. We expect to rule out parts of those $\mathcal{R}_k$ for $k \in A_J$, but this expectation can be overturned, if the simulator is non-linear or has interactions. $\mathcal{R}_k$ can always be extended, rather than shrunk, if the NROY runs are piling up against the lower or upper boundary. Then go back to the previous section and initiate the next wave with a new observable, or set of observables.

A more sophisticated approach, appropriate when the ensemble of runs

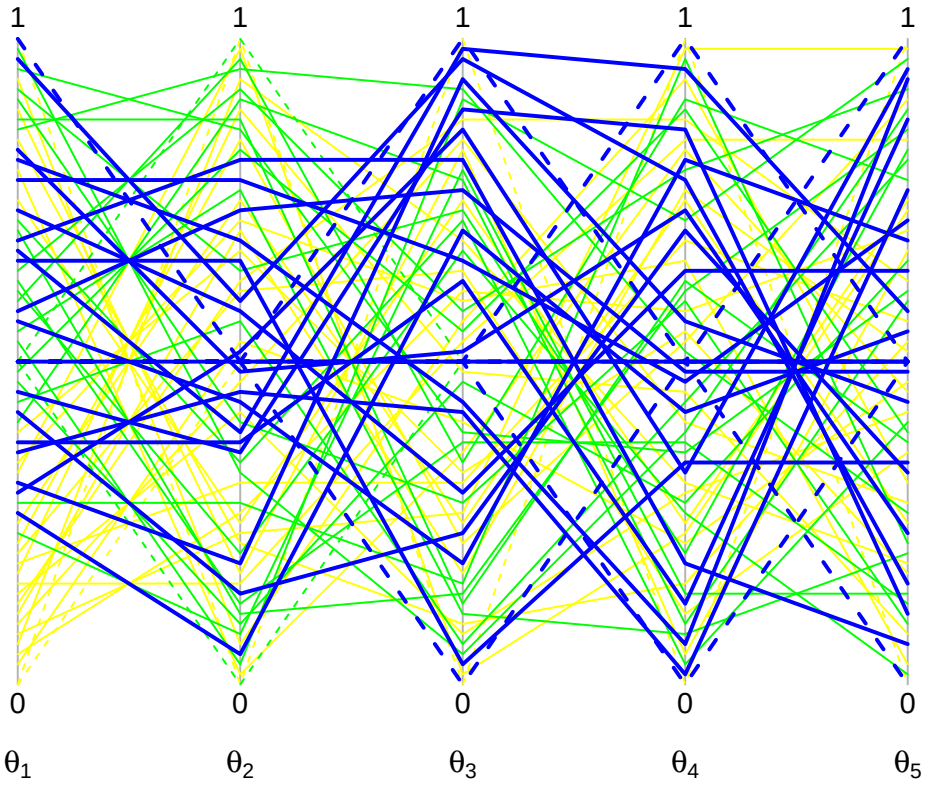


Figure 3: Parallel Coordinates Plot (PCP) for assessing $\mathcal{C}_{Jk}$ for each parameter $k = 1, \ldots, 5$. Each parameter has its own vertical axis, and each run is represented as a line joining the axes. The colour of each line represents the implausibility, with $I_J^{(i)} \leq 3$ = dark-blue, 3–6 = green, $> 6$ = yellow. I use darker colours and thicker lines for lower implausibilities, and set the plotting order of the lines from high to low implausibility, so that the NROY runs stand out. This PCP is drawn for 13 sanity check runs plus 60 runs in a Sobol sequence, using $J \leftarrow \{1\}$ and the observation $y_1^{\text{obs}}$ from the example in the text (see section 4 and the end of section 6). The sanity check runs are shown with dashed lines.

inside the current $\mathcal{R}$ is large, is to use a non-linear response surface in computing $v_J$ in (9), and in choosing between alternative candidates for $J$. The crudely-constructed $\pi_{jk}$ table from section 6 is used for generating candidates for $J$, but not for choosing between them. This table is useful in the absence of strong beliefs about the simulator, but strong beliefs should take precedence, if they exist. Note that it is quite consistent to have a strong belief about $\theta_k$ being an active parameter for $y_j$, while at the same time being very uncertain about $\mathcal{R}_k$.

Section 9 (Last gasp design) would follow on naturally at this point, but the following material on reparameterisation is more important.

## 8 Reparameterising fields

I am going to assume that you have your reasons for specifying the parameters the way you have. You have chosen $(\theta_i, \theta_j)$ rather than, say, $(\theta_i + \theta_j, \sqrt{\theta_i})$ because $\theta_i$ and $\theta_j$ are somewhat meaningful to you. Therefore, you are not particularly interested in transformations of the parameters.

There is one obvious exception, though, which is when a subset of the parameters is a field of similar values, such as Manning's $n$ values for the reaches of a channel, or permeabilities for the voxels of a 3D hydrocarbon reservoir simulator. The problem here is that space-filling designs take a very long time to visit all of the corners of $\mathcal{R}$. Therefore, the range of values for the arithmetic mean of a field of parameters from a space-filling design will typically be much less than the range of the individual parameters. Unfortunately, the amount of variation in the mean depends on the number of runs in the design and the number of parameters in the field: it is not under the direct control of the experimenter.

This issue is important if the mean of the field is uncertain. Furthermore, it often happens that there is an observable which is sensitive to the mean. For example, flow in a channel may be measured at a particular reach, but it is sensitive to all of the Manning's $n$ values in the channel, not just the one at the reach. In this situation it is crucial to be able to vary the mean of the field within a specified range, and to be able to revise this range using

observations.

I generalise slightly, to allow for the field to be divided into $k$ regions, where the mean of the field in each region is to be varied within a specified range: this is the case in hydrocarbon reservoir simulators, where the regions are defined by geological features such as faults. Let there be $p$ parameters in the field, with $p \gg k$, and let $A$ be a non-negative $k \times p$ matrix satisfying $A\mathbf{1}_p \leq \mathbf{1}_p$, where $\mathbf{1}_p$ is the $p$-dimensional vector of 1's. So $A\theta$ would typically be the vector of regional means, with $A$ satisfying $A\mathbf{1}_p = \mathbf{1}_p$. Assume, without loss of generality, that the range of each parameter is $(0, 1)$, so that the range of each $A\theta$ must be contained in $(0, 1)$.

For an $n$-run design, let $X$ be a $n \times p$ design matrix for values of $\theta$, and let $Z$ be a $n \times k$ matrix of values for $A\theta$. $X$ will likely be a space-filling design on $(0, 1)^{n \times p}$; $Z$ might also be a space-filling design on $(0, 1)^{n \times k}$, if it is desired to take the means to their lowest and highest possible values, but it might also be space-filling on narrower ranges. Simple linear algebra can be used to nudge each row of $X$ to hit the target value from $Z$. If $x^T$ and $z^T$ are the rows, then

$$x' \leftarrow x - A^T(AA^T)^{-1}(Ax - z), \tag{12}$$

where $x'$ is the nudged value (premultiply by $A$ to check that $Ax' = z$). Readers familiar with spatial statistics will recognise this approach as conditioning by Kriging, see Rue and Held (2005, sec. 2.3.3).

Some of the $x'$ values may lie outside the parameter space $(0, 1)^p$. This can be fixed by transforming each element of $x$ and $z$ to the real line using the Probit transformation, computing $x'$, and then transforming back. Although this does not exactly ensure that $Ax' = z$ (the transformation being non-linear), the main objective will have been achieved, which is to substantially increase the range of $A\theta$ compared to its range in a space-filling design.

In the pre-wave analysis of section 6 and in the PCPs of section 7, the components of $\theta$ that correspond to the field should be replaced by $A\theta$'s. In other words, replace $p$ parameters with $k$ transformed parameters, where $k \ll p$. With any luck some observations will revise the ranges of some of the $A\theta$'s. These revised ranges can be used in constructing the $Z$ matrix for the

next wave, while also preserving the full range of the individual parameters.

Figures 4 and 5 show some of a 100-point Sobol sequence over 20 parameters, without and with the nudging. The lack of variation in the original mean is very clear in the final column of Figure 4. This has been corrected by the nudging, in Figure 5. It is true that the nudged design is not as beautiful as the original, and that it is a bit clumpier. But the original design is basically useless as far as varying the mean is concerned, and the effect would be even worse for a design with fewer runs, or a field with a larger number of components.

## 9 Last gasp design

There may come a time, as you approach the end of setting-up, when you feel the urge to generate a final wave of runs that are mainly NROY. Understandably, you would like to push at the boundaries of the NROY region from the inside. There is a simple strategy for this, which relies on you having kept all of the runs you have done so far, *including those that were* `NA`. Standardise the parameter values for the complete set of runs to $(0, 1)^p$ using the minimum and maximum values for each parameter. Then fill $(0, 1)^p$ with a space-filling design. Compute the Euclidean distance between each point in the space-filling design and each point in the complete set of runs. Then discard all of the points in the space-filling design that are closer to an implausible point (including `NA`) than to an NROY point. Then run the ones that are left, or a space-filling subset of them, if it is too expensive to run them all.

Figure 6 shows the result of doing a last gasp design after just one wave of the toy problem. One thousand points were used in a space-filling design, of which 349 were closer to an NROY run than implausible one. These were reduced to 30 points by finding a subset with a large minimum interpoint distance (the maximin criterion). These 30 were run, and the PCP of the result is shown in Figure 6, on the original scale. 12 of these runs are NROY. When added to the 26 from the sanity check and first wave, this makes a pool of 38 sets of parameter values that are NROY. In terms of proportions,

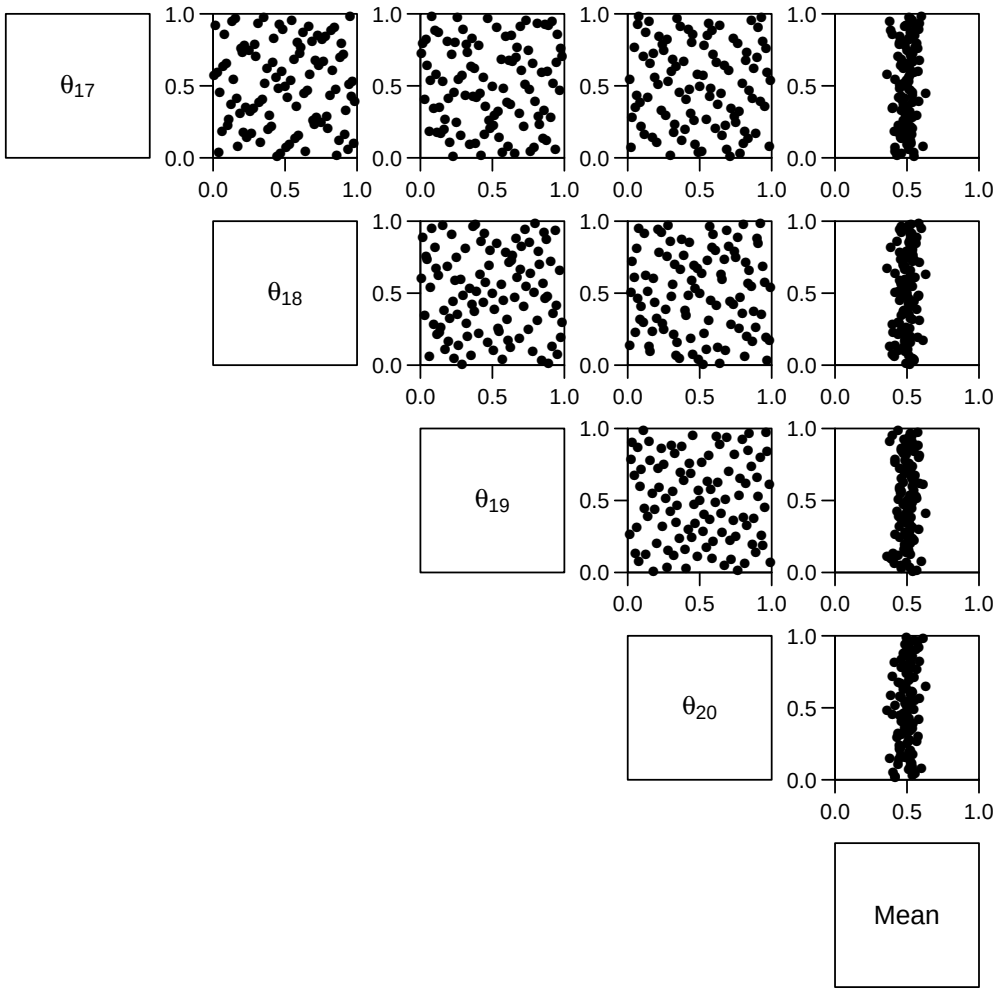


Figure 4: Bottom corner of a pairs plot for a 100-point Sobol sequence in 20 dimensions, plus the mean value (final column).

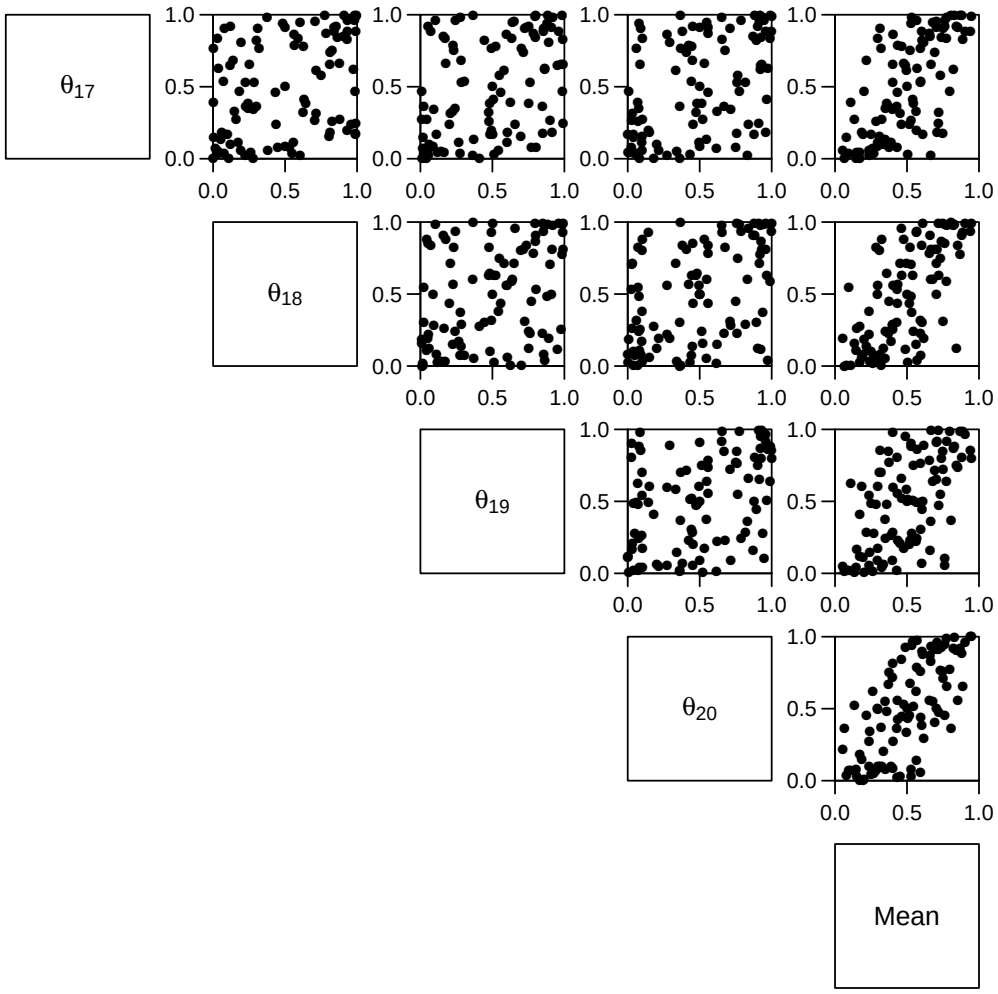


Figure 5: Same as Figure 4, except after nudging the points to get better coverage of the mean; see section 8 for details.

the last gasp design has $12/30 = 0.40$ of NROY runs, while the sanity check and first wave together have $26/73 \approx 0.36$ of NROY runs. The proportion of NROY runs in any follow-up design can be made arbitrarily close to 1, simply by shadowing existing NROY runs. The last gasp design is designed to fill the space around the NROY runs, and therefore there is always the possibility that some of its runs will be implausible: this is the price of a space-filling design.

Figure 7 shows the PCP for all 103 runs. There is a small expansion to be made at the upper end of $\mathcal{R}_2$, from 0.81 to 0.85. Because the last gasp runs push at the NROY region from the inside, it is to be expected that the parameter ranges might expand slightly. Again, it is worth reiterating that this reduction in size of both $\mathcal{R}_1$ and $\mathcal{R}_2$ on the basis of one observable is a good outcome.

## 10 Stochastic simulators

A stochastic simulator is a deterministic simulator with an additional input: the seed of a numerical random number generator. Hence a stochastic simulator can be written as the deterministic simulator

$$f(\theta, \omega_0)$$

where $\omega_0$ is the seed. In a computer, $\omega_0$ expands out to the deterministic sequence $\omega_0, \omega_1, \ldots, \omega_m$, and so a stochastic simulator can also be written as the deterministic simulator

$$f(\theta, \omega_0, \omega_1, \ldots, \omega_m).$$

It is the constructor of the simulator who has decided to collapse the whole sequence of $\omega$'s down to a single arbitrary value $\omega_0$. This must follow from her belief that the available observations are not able to constrain the individual $\omega$'s in a useful way. Thus $\omega_0$ represents a source of uncertainty, but not an opportunity for uncertainty reduction.

There are three main uses for a stochastic sequence in a simulator, all of

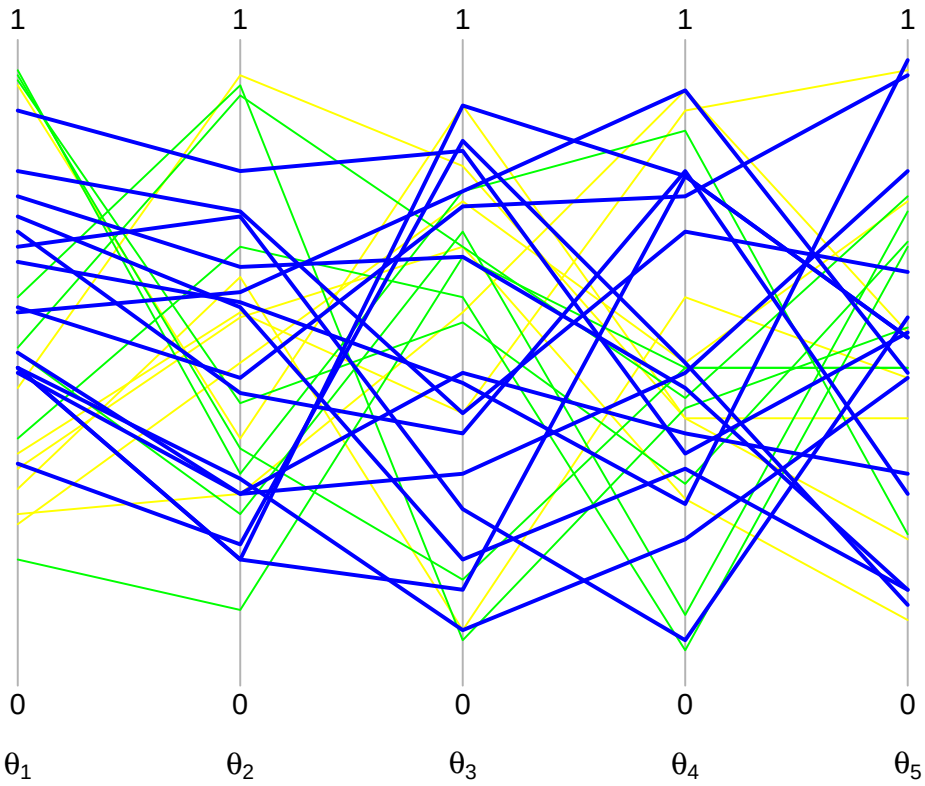


Figure 6: PCP of 30 last gasp design runs (section 9). See Figure 3 for details of the PCP.

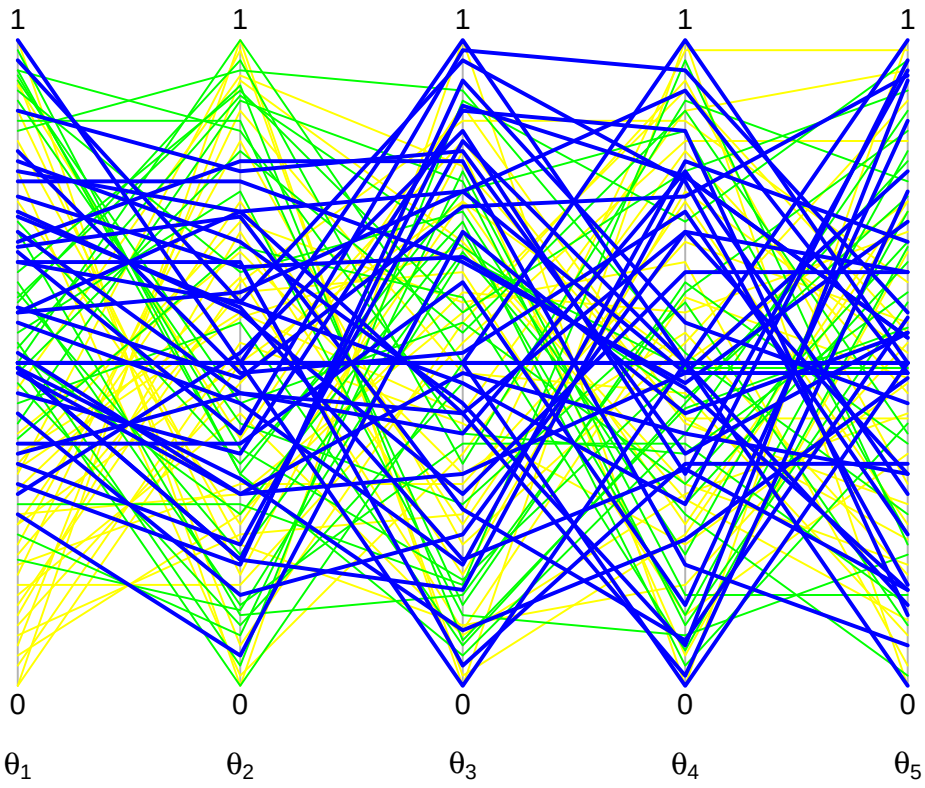


Figure 7: PCP for all 103 runs, combining the runs in Figures 3 and 6.

which can be exemplified in a dyamical simulator for $x(t)$,

$$\mathrm{d}x = f_t(x;\theta)\,\mathrm{d}t$$

in its deterministic form. First, the system might include a high-dimensional process that is best captured by a stochastic process. In a climate simulator, for example, this might be the effect of forcings such as volcanism or solar insolation; in hydrology it might be precipitation on the catchment, at the pixel scale. This high-dimensional process can be added to the deterministic form of the simulator as

$$\mathrm{d}x = f_t(x;\theta)\,\mathrm{d}t + \tau\,\mathrm{d}\xi \tag{13}$$

where $\xi$ is a standardised stochastic process. Its scale $\tau$ can be treated as an additional parameter, or it can be specified if the properties of the process are known.

Second, the stochastic sequence can provide a more nuanced treatment of structural uncertainty. The form would be the same as (13), although in this case $\tau\,\mathrm{d}\xi$ would be the correction from the deterministic simulator value

$$x(t+\mathrm{d}t) = x(t) + f_t(x;\theta)\,\mathrm{d}t$$

to the system value. In this case $\tau$ is more likely to be unknown, and treated as an additional parameter, and $\xi$ might also be a more complex process than simple Brownian motion. Penland (2007) gives a more detailed account of this approach, while Crucifix and Rougier (2009) is an implementation for a low-order simulator of ice-ages (see also Rougier, 2013). In this use, the value $\sigma^2_{\mathrm{str},j}$ in (2) can be set to zero, as it will be replaced by the value described below.

Third, a stochastic initial condition plus a spin-up can be used to initialise a deterministic dynamical simulator with a stochastic $x_0$, particularly in a simulator with sensitive dependence on initial conditions.

When a stochastic differential equation is solved numerically, for example using a stochastic predictor-corrector scheme (see Burrage *et al.*, 2004), the

result is one *replication*. Averaging the simulator observables over $r$ replications, involving $r$ different values for $\omega_0$, gives the sample mean vector $\bar{y}(\theta)$ and the sample variance matrix $S(\theta)$. These are estimates of the true values (no scare quotes), which could be determined to arbitrary precision with an unlimited number of replications.

For computing implausibility, $y_j(\theta)$ in (2) needs to be replaced by $\bar{y}_j(\theta)$, and the variance $\sigma_j^2$ needs to include the additional term $S_{jj}(\theta)(1 + 1/r)$ if there are $r$ replicates. To understand this additional term, imagine that a separate very accurate estimate of the variance were available, say $T(\theta)$. If the expectation of $y_j(\theta)$ is used in the numerator of (2), then $\sigma_j^2$ should include $T_{jj}(\theta)$ in the denominator, to account for the extra uncertainty from the stochastic term. But if the expectation in the numerator is replaced by the sample mean $\bar{y}_j(\theta)$, then there is an additional source of uncertainty, which would give $T_{jj}(\theta) + T_{jj}(\theta)/r$ in the denominator. If we do not have $T(\theta)$ then it must be replaced by $S(\theta)$, and we end up with $S_{jj}(\theta)(1 + 1/r)$. Of course, the extra $1/r$ may not matter, given all of the other terms in $\sigma_j^2$.

With small numbers of replicates it makes sense to use a smoothed estimate for $S(\theta)$, such as a pooled estimate over a set of similar observables. A pilot study at the start of the experiment might show that $S(\theta)$ is relatively insensitive to $\theta$ (in terms of its contribution to the $\sigma_j^2$'s), in which case pooling over different values of $\theta$ is also possible. If this pooled estimate is $T$, then the additional term in $\sigma_j^2$ is $T_{jj}(1 + 1/r)$.

Rougier and Goldstein (2014) consider a related situation which arises in deterministic dynamical simulators with attractors, such as climate simulators. In this case dimensional reduction and time-series methods could be used to summarise the attractor in terms of a $\bar{y}(\theta)$ and $S(\theta)$, even though the simulator itself is deterministic.

## 11 Emulators

What happens if the number of runs in a wave exceeds the budget of CPU cycles or time? An *emulator* offers a partial solution. An emulator is a statistical model of the simulator, and the idea is to use as many runs as

you can afford (in a space-filling design) to build the emulator for the chosen observables, and then use the emulator in place of the simulator to compute the implausibility. The use of emulators for history matching is described in Vernon *et al.* (2010).

The key thing to appreciate is that building an emulator is usually an act of desperation. The simulator observables are likely to be a more complex function of $\theta$ than an emulator can reasonably represent, and the limited number of runs cannot adequately represent the complexity of the simulator. In this situation, the advantage of a statistical emulator over a less-statistical approach (such as a neural network) is that it attempts to represent its uncertainty about the simulator output at $\theta$. Thus the emulator predicts the observable $y_j(\theta)$ in terms of an expectation $\mu_j(\theta)$ and a variance $\Sigma_{jj}(\theta)$. For simplicity, I will assume that the simulator is deterministic. Then $\mu_j(\theta)$ is an interpolator of the runs in a neighbourhood of $\theta$, and $\Sigma_{jj}(\theta)$ is the expected squared interpolation error. Emulators also extrapolate outside the convex hull of the runs, although $\Sigma_{jj}(\theta)$ is less reliable when extrapolating.

The form of $\mu$ and $\Sigma$ are learnt from the available runs, typically after removing legacy runs which lie outside the current $\mathcal{R}$, although some on the edge might be retained to reduce extrapolation. Emulators built by conditioning Gaussian processes are consistent in the sense that if the parameter value $\theta$ has been run, then $\mu(\theta') \to f(\theta)$ and $\Sigma(\theta') \to \mathbf{0}$ as $\theta' \to \theta$. This is a very attractive property, if the simulator is a smooth function of the parameters. Gaussian process are described in Rasmussen and Williams (2006) and Forrester *et al.* (2008). Software for Gaussian process emulation is provided by Roustant *et al.* (2012). Rougier (2008) describes a tractable multivariate gaussian process emulator, and Rougier *et al.* (2009a) illustrate some of the issues that arise when emulating a large and complex simulator.

For computing implausibility, $y_j(\theta)$ in (2) needs to be replaced by $\mu_j(\theta)$, and the variance $\sigma_j^2$ needs to include the additional term $\Sigma_{jj}(\theta)$. The presence of $\Sigma_{jj}(\theta)$ flattens the implausibility surface and enlarges the NROY region, compared to the case where the simulator can be run directly. On the other hand, the emulator $\mu$ and $\Sigma$ are very fast to evaluate, and this means that the projection operation from $I_J(\theta)$ to $I_J(\theta_k)$ can be done explicitly, rather

than approximately using a thick slice through the runs, as was described in section 7. So the PCP for an emulator can have thousands of lines, if the emulator is used to evaluate implausibility on a high-resolution space-filling design in the parameter space.

With a well-built emulator we have, in effect, an accurate determination of a superset of the NROY region, compared to an inaccurate determination of the actual NROY region. In the best possible outcome the emulator prediction error $\Sigma_{jj}$ is small across the parameter space (relative to the other terms in the $\sigma_j$'s), and then an emulator improves on the outcome from using the runs directly, giving an accurate determination of the NROY region. But this must be set against the cost of building and checking the emulator for the observables in $J$, which is not (yet) an automated process, since there are many subjective and influential choices to be made (see, e.g., Rougier *et al.*, 2009a). An emulator is a complex statistical model, and constructing an emulator is a task for a statistician, working in conjunction with system experts.

## Acknowledgements

The list of people who have helped me to develop my understanding of computer experiments would stretch to several pages—if I could remember everyone! But I cannot not mention my former colleagues at Durham (UK), notably Peter Craig, Michael Goldstein, and Allan Seheult, and the 'new generation' of history matchers, notably Ian Vernon and Daniel Williamson. I must also mention Tamsin Edwards, who has been an exemplary collaborator on several very challenging computer experiments. For this paper I would like to thank Keith Beven in particular for his perceptive comments on a previous draft, and also for the many discussions we have had about the intersection of statistics and environmental modelling; I must also absolve him of some of the views expressed here. This work was supported by the Natural Environment Research Council (NERC) funded Consortium on Risk in the Environment: Diagnostics, Integration, Benchmarking, Learning and Elicitation (CREDIBLE); grant number NE/J017450/1.

# A Glossary

**Active parameters** Those parameters whose values are the main determinants of the value of a specified output/observable.

**Boundary values** Fixed features of the system which are an input to the simulator.

**Calibration** Probabilistic conditioning of the parameters (and other inputs) on observations.

**Diary management problem (DMP)** Setting-up a simulator is about making the best use of your time, as well as your computing resources.

**Emulator** A statistical model of the simulator, constructed from an ensemble of runs.

**Forcing** Variable (typically in time) features of the system which are an input to the simulator.

**History matching** Proceeding by ruling out regions of the parameter space according to the disagreement between the simulator observables and the observations.

**Implausible** A parameter value for which a simulator observable is too far away from the corresponding observation.

**Inputs** Forcing, boundary, parameters, and initial values, all requiring to be specified before the simulator will run.

**Legacy runs** Runs from previous waves whose parameter values are no longer in the current parameter ranges.

**Model** Reserved for the respresentation of the system, on which the simulator is based.

**Non-behavioural** Parameter values for which the simulator output is inconsistent with our understanding of system behaviour. Sufficient for implausible.

**NROY** Not Ruled Out Yet, of a point in the parameter space; a point with a currently low implausibility.

**Observables** Known functions of the simulator outputs which correspond to operationally-defined system values.

**Observations** Measured system quantities corresponding to observables.

**Outputs** The values produced by the simulator for specified inputs.

**Parameters** Ajustable coefficients in the simulator which tend to correspond to non-measurable aspects of the system.

**Replications** Multiple runs of a stochastic simulator at the same input values (by changing the value of the random seed).

**Representation error** Incommensurability between the simulator observables and the observations.

**Setting-up** Identifying the key parameters, and setting/adjusting their ranges. The subject of this paper.

**Simulator** The computer code which maps inputs into outputs. If two runs of the simulator at the same inputs do not necessarily produce the same outputs, the simulator is 'stochastic', otherwise it is 'deterministic'.

**Spin up** Running a dynamical simulator from an earlier starting point in order to replace the choice of initial value with a less non-behavioural one.

**Sticky run** A run that takes a long time to complete, often because of adaptive time-stepping in a dynamical simulator.

**Structural error** The mismatch between the simulator outputs and the system values that cannot be tuned away given knowledge of forcing, boundary, and initial values.

**System** The process under study.

**Tuning** Iteratively adjusting the parameters (and possibly other inputs) to improve the agreement between the simulator observables and the observations.

**Wave** One iteration of the process of running the simulator multiple times, and then analysing the results in terms of revising the parameter space.

## References


K.J. Beven, 2009. *Environmental Modelling; An Uncertain Future*. Routledge, Abingdon, Oxon, UK.

K. Burrage, P.M. Burrage, and T. Tian, 2004. Numerical methods for strong solutions of stochastic differential equations: An overview. *Proceedings of the Royal Society of London, Series A*, **460**, 373–402.

G. Casella and R.L. Berger, 2002. *Statistical Inference*. Pacific Grove, CA: Duxbury, 2nd edition.

D.R. Cox and D.V. Hinkley, 1974. *Theoretical Statistics*. London: Chapman and Hall.

P.S. Craig, M. Goldstein, A.H. Seheult, and J.A. Smith, 1996. Bayes linear strategies for matching hydrocarbon reservoir history. In J.M. Bernado, J.O. Berger, A.P. Dawid, and A.F.M. Smith, editors, *Bayesian Statistics 5*, pages 69–95. Oxford: Clarendon Press.

P.S. Craig, 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. In C. Gatsonis, J.S. Hodges, R.E. Kass, R. McCulloch, P. Rossi, and N.D. Singpurwalla, editors, *Case Studies in Bayesian Statistics III*, pages 37–87. New York: Springer-Verlag. With discussion.

M. Crucifix and J.C. Rougier, 2009. On the use of simple dynamical systems for climate predictions: A Bayesian prediction of the next glacial inception. *The European Physics Journal – Special Topics*, **174**(1), 11–31.

N. Edwards, D. Cameron, and J.C. Rougier, 2011. Precalibrating an intermediate complexity climate model. *Climate Dynamics*, **37**, 1469–1482.

P.W. Fitzgerald, J.L. Bamber, J. Ridley, and J.C. Rougier, 2011. Exploration of parametric uncertainty in a surface mass balance model applied to the Greenland ice sheet. *Journal of Geophysical Research*, **117**, F01021.

A.I.J. Forrester, A. Sóbester, and A.J. Keane, 2008. *Engineering Design via Surrogate Modelling: A Practical Guide*. Chichester, UK: John Wiley & Sons.

M. Goldstein and J.C. Rougier, 2004. Probabilistic formulations for transferring inferences from mathematical models to physical systems. *SIAM Journal on Scientific Computing*, **26**(2), 467–487.

M. Goldstein and J.C. Rougier, 2009. Reified Bayesian modelling and inference for physical systems. *Journal of Statistical Planning and Inference*, **139**, 1221–1239. With discussion, pp. 1243–1256.

N.M. Hunter, P.D. Bates, M.S. Horritt, and M.D. Wilson, 2007. Simple spatially-distributed models for predicting flood inundation: A review. *Geomorphology*, **90**, 208–225.

M. Maggioni *et al.*, 2013. A new experimental snow avalanche test site at Seehore peak in Aosta Valley (NW Italian Alps)–part I: Conception and logistics. *Cold Regions Science and Technology*, **85**, 175–182.

J. Murphy, R. Clark, M. Collins, C. Jackson, M. Rodwell, J.C. Rougier, B. Sanderson, D. Sexton, and T. Yokohata. Perturbed parameter ensembles as a tool for sampling model uncertainties and making climate projections. In *Proceedings of ECMWF Workshop on Model Uncertainty, 20-24 June 2011*, pages 183–208, 2011. Available online, `http://www.ecmwf.int/publications/library/ecpublications/_pdf/workshop/2011/Model_uncertainty/Murphy.pdf`.

J.C. Neal, P.D. Bates, T.J. Fewtrell, N.M. Hunter, M.D. Wilson, and M.S. Horritt, 2009. Distributed whole city water level measurements from the Carlisle 2005 urban flood event and comparison with hydraulic model simulations. *Journal of Hydrology*, **368**, 42–55.

C. Penland, 2007. Stochastic linear models of nonlinear geosystems. In A.A. Tsonis and J.B. Elsner, editors, *Nonlinear Dynamics in Geosciences*, pages 485–515. New York: Springer.

F. Pianosi *et al.*, 2015. Sensitivity analysis of environmental models: A systematic review with practical workflow. In submission.

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

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013.

C.E. Rasmussen and C.K.I. Williams, 2006. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge MA, USA. Available online at `http://www.GaussianProcess.org/gpml/`.

C.P. Robert and G. Casella, 2004. *Monte Carlo Statistical Methods*. Springer, New York NY, 2nd edition.

J.C. Rougier, 2007. Probabilistic inference for future climate using an ensemble of climate model evaluations. *Climatic Change*, **81**, 247–264.

J.C. Rougier, 2008. Efficient emulators for multivariate deterministic functions. *Journal of Computational and Graphical Statistics*, **17**(4), 827–843.

J.C. Rougier, 2013. 'Intractable and unsolved': Some thoughts on statistical data assimilation with uncertain static parameters. *Phil. Trans. R. Soc. A*, **371**, 20120297.

J.C. Rougier and K.J. Beven, 2013. Model and data limitations: The sources and implications of epistemic uncertainty. In Rougier *et al.* (2013b), chapter 3.

J.C. Rougier and M. Goldstein, 2014. Climate simulators and climate projections. *Annual Review of Statistics and Its Application*, **1**, 103–123.

J.C. Rougier, M. Goldstein, and L. House, 2013a. Second-order exchangeability analysis for multi-model ensembles. *Journal of the American Statistical Association*, **108**, 852–863.

J.C. Rougier, S. Guillas, A. Maute, and A. Richmond, 2009a. Expert knowledge and multivariate emulation: The Thermosphere-Ionosphere Electrodynamics General Circulation Model (TIE-GCM). *Technometrics*, **51**(4), 414–424.

J.C. Rougier and M. Kern, 2010. Predicting snow velocity in large chute flows under different environmental conditions. *Applied Statistics*, **59**(5), 737–760.

J.C. Rougier, D.M.H. Sexton, J.M. Murphy, and D. Stainforth, 2009b. Analysing the climate sensitivity of the HadSM3 climate model using ensembles from different but related experiments. *Journal of Climate*, **22** (13), 3540–3557.

J.C. Rougier, R.S.J. Sparks, and K.V. Cashman, 2015. Global recording rates for large eruptions. In submission, available from the first author, email `j.c.rougier@bristol.ac.uk`.

J.C. Rougier, R.S.J. Sparks, and L.J. Hill, editors, 2013b. *Risk and Uncertainty Assessment for Natural Hazards*. Cambridge University Press, Cambridge, UK.

O. Roustant, D. Ginsbourger, and Y. Deville, 2012. DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by Kriging-based metamodelling and optimization. *Journal of Statistical Software*, **51**(1), 1–55. `http://www.jstatsoft.org/v51/i01/`.

H. Rue and L. Held, 2005. *Gaussian Markov Random Fields: Theory and Applications*, volume 104 of *Monographs on Statistics and Applied Probability*. Chapman & Hall/CRC, Boca Raton FL, USA.

F. Tiefenbacher and M. Kern, 2004. Experimental devices to determine snow avalanche basal friction and velocity profiles. *Cold Regions Science and Technology*, **38**, 17–30.

W.N. Venables and B.D. Ripley, 2010. *Modern Applied Statistics with S*. Springer, New York NY, USA, fourth edition.

I. Vernon, M. Goldstein, and R.G. Bower, 2010. Galaxy formation: A Bayesian uncertainty analysis. *Bayesian Analysis*, **5**(4), 619–670.

A. Zammit-Mangion, J.C. Rougier, J. Bamber, and N. Schön, 2014. Resolving the Antarctic contribution to sea-level rise: A hierarchical modelling framework. *Environmetrics*, **25**, 245–264.