NISS

A Framework for Validation of Computer Models

M. J. Bayarri, J. O. Berger, R. Paulo, J. Sacks,
J. A. Cafeo, J. Cavendish, C. H. Lin, J. Tu

Technical Report Number 162
April 2005

National Institute of Statistical Sciences
19 T. W. Alexander Drive
PO Box 14006
Research Triangle Park, NC 27709-4006
www.niss.org

A Framework for Validation of Computer Models[*]

M.J. Bayarri, J.O. Berger, R. Paulo, J. Sacks
National Institute of Statistical Sciences

J.A. Cafeo, J. Cavendish, C.H. Lin, J. Tu
General Motors

April 23, 2005

Abstract

In this paper, we present a framework that enables computer model evaluation oriented towards answering the question:

Does the computer model adequately represent reality?

The proposed validation framework is a six-step procedure based upon a mix of Bayesian statistical methodology and likelihood methodology. The methodology is particularly suited to treating the major issues associated with the validation process: quantifying multiple sources of error and uncertainty in computer models; combining multiple sources of information; and updating validation assessments as new information is acquired. Moreover, it allows inferential statements to be made about predictive error associated with model predictions in untested situations.

The framework is illustrated on two test bed models (a pedagogic example and a resistance spot weld model) that provide context for each of the six steps in the proposed validation process.

[*]David Higdon and Marc Kennedy were central to the development of an earlier version of this framework.

Contents

1 Introduction

1.1 Motivation and overview

We view the most important question in evaluation of a computer model to be

Does the computer model adequately represent reality?

An austere view expressed in Oreskes et al. (1994) is that validating a computer model cannot be done and that the primary value of models is heuristic: models are representations, useful for guiding further study but not susceptible to proof. This view has substantial basis in purely scientific roles, as distinct from a models use in policy and engineering contexts. But the real question, we contend, is not whether a model is absolutely correct or only a useful guide. Rather, it is to assess the degree to which it is an effective surrogate for reality: does the model provide predictions accurate enough for intended use?

The question and attitude we set is not new. It appears over and again in discussions and comments on validation in many arenas over the years, at least as long ago as Caswell (1976). A detailed discussion of many issues surrounding validation can be found in Berk et al. (2002). But, incisive argument on the validity of models, seen as assessment of their utility, has been hampered by the lack of structure in which quantitative evaluation of a models performance can be addressed. It is our purpose here to explore structure and methodology to produce such evaluations.

In practice, the processes of computer model development and validation often occur in concert; aspects of validation interact with and feed back to development (e.g., a shortcoming in the model uncovered during the validation process may require change in the mathematical implementation). In this paper, however, we address the process of computer model development only to the extent that it interacts with the framework we envision for evaluation; the bulk of the paper focuses on answering the basic question posed above. In particular, we do not address the issue of code verification. General discussions of the entire Validation and Verification process can be found in Roache (1998), Oberkampf and Trucano (2000), Cafeo and Cavendish (2001), Easterling (2001), Pilch et al. (2001), Trucano et al. (2002), and Santner et al. (2003).

Tolerance bounds

To motivate the approach we take to model evaluation, it is useful to begin at the end, and consider the type of conclusions that will result from the methodology. As noted above we do not focus on answering the yes/no question "Is the model correct?" but rather on assessing accuracy of predictions in uses of the model. This will be done by presenting *tolerance bounds,* such as 5.17 ± 0.44, for a model prediction 5.17, with the interpretation that there is a specified chance (e.g., 90%) that the corresponding true process value would lie within the specified range. Such tolerance bounds should be given *whenever predictions are made*, i.e., they should routinely be included along with any predictions arising from use of the model.

This focus on giving tolerance bounds, rather than stating a yes/no answer as to model validity, arises for three reasons:

1. Models rarely give highly accurate predictions over the entire range of inputs of possible interest, and it is often difficult to characterize regions of accuracy and inaccuracy.
2. The degree of accuracy that is needed can vary from one application of the computer model to another.
3. Tolerance bounds account for *model bias*, the principal symptom of model inadequacy; accuracy of the model cannot simply be represented by a variance or standard error.

All these difficulties are obviated by the simple device of routinely presenting tolerance bounds along with model predictions. Thus, at a different input value, the model prediction and tolerance bound might be 6.28 ± 1.6, and it is immediately apparent that the model is considerably less accurate at this input value than at the previous input, where the tolerance bound was ± 0.44. Either of the bounds, 0.44 or 1.6, might be acceptable or unacceptable, depending on the intended use of the model.

Producing tolerance bounds is not easy. Here is a partial list of the hurdles one faces:

- There are uncertainties in model inputs or parameters, and these uncertainties can arise in several ways: based on data, expert opinion, or simply a prior 'uncertainty range.'
- When model runs are expensive, only limited model-run data may be available.
- Field data of the actual process under consideration may be limited and noisy.
- Output data may be of a variety of types, including functional data.
- Model runs may be made at input values different from those at which field data are observed.
- One may desire to 'tune' unknown parameters of the computer model based on field data, and at the same time (because of sparse data) apply a validation methodology.
- There may be more tuning parameters than data.
- The computer model itself will typically be highly non-linear.
- Accounting for possible model bias is challenging.
- Validation should be viewed as an accumulation of evidence to support confidence in the model outputs and their use, and the methodology should allow updating current conclusions as additional information arrives.

This paper describes an approach that deals with these hurdles and, utilizing a mix of Bayesian and likelihood techniques, can produce usable tolerance bounds for computer model predictions, thereby giving specific quantitative meaning to validation. Technical details of the approach are given in Section 5 while the remainder of this section is given over to added discussion of validation and the approach we recommend. Details addressing some of the listed hurdles are not given because they are not needed for the testbed problems. For example, uncertainties in inputs based on data are not addressed but can be accommodated in a straightforward way.

Bridging two philosophies

At the risk of considerable oversimplification, it is useful to categorize approaches to model evaluation as being in two camps. In one, evaluation is performed primarily by comparing model output to field data from the real process being modelled. The rationale for this *predictive* approach is simple: the only way to see if a model actually works is to see if its predictions are correct.

The second camp focuses primarily on the model itself and tries to assess the accuracy or uncertainty corresponding to each constructed element of the model. The rationale for this *physical* approach is that, if all the elements of the model (including computational elements) can be shown to be correct, then logically the model must give accurate predictions.

Our view lies in the predictive camp. Without demonstration of validity on actual field data a modeller faces great difficulty in convincing others that all elements of the model are correctly constructed. Insufficient field data may drive modellers to focus on the physical approach but the lack of field data will always leave the model suspect.

Of course a Bayesian approach bridges both philosophies through informativeness of the priors. The physical approach assumes a highly informative prior, the predictive approach (the one we take) relying more on non-informative priors, even while taking advantage of information about elements of the model.

Side benefits of the methodology

Implementation of the suggested methodology has these added implications:

1. The methodology allows explicit estimation of the bias of the model (together with the uncertainty in the bias), through comparison with field data. This allows direct judgement as to the validity of the model in various regions of the input space. In addition, the methodology allows one to adjust the prediction by the estimated bias, and provides tolerance bounds for this adjusted prediction. Depending on the size of the bias this can result in considerably more accurate predictions than use of the model alone (or use of field data alone). Note, however, that this adjustment might have limited utility in extrapolation to new situations, unless one is willing to make strong assumptions about how the bias extrapolates.

2. Predictions and tolerance bounds can be given for applications of the computer model to new situations in which there are little, or no, field data, assuming information about 'related' scenarios is available; this can be done through hierarchical Bayesian analysis. We do not address this in the current paper; the predictions below in the testbed examples are made within the context of the given scenarios.

3. Fast approximations to the computer code are used (typically, needed) for the proposed methods; these approximations have additional utility for use with complex computer codes in other contexts – such as in optimization.

Background and Applicability

The key components of the approach outlined here are the use of Gaussian process response-surface approximations to a computer model, following on work in Sacks et al. (1989), Currin et al. (1991), Welch et al. (1992), and Morris et al. (1993), and introduction of Bayesian representations of model bias and uncertainty, following on work in Kennedy and O'Hagan (2001) and Kennedy et al. (2002). The Gaussian process approximations have proved valuable in real settings where functions are complex and data are limited (for example, Gough and Welch (1994), Chapman et al. (1994), Aslett et al. (1998)) and their adoption in the methodology we formulate is both natural and convenient.

The main motivation for this paper is to outline the step-by-step process that was developed for engineers to produce tolerance bounds that take into account the key uncertainties in the problem. The process is illustrated in the paper on a simple pedagogical example – for which we know the truth – and on an engineering example involving spot-welding.

The approach taken here results in a computational burden that significantly increases with large numbers of model inputs, large numbers of unknown parameters, or a large amount of data (model-run or field). Hence a primary concern is to focus on methods that have the potential to significantly scale-up.

Originally, as described in the companion paper Higdon et al. (2004), a fully Bayesian approach to the problem was developed. But this has difficulties in appropriately scaling up, and also requires considerable expertise in MCMC computation. Hence we have focused instead on simplifications such as 'modularity' (analyze components of the problem separately to the extent possible), and use of maximum likelihood or other methods to reduce the computational burden and allow the Bayesian part of the analysis to be stable.

Validation is intrinsically a hard statistical problem and analyses that produce tolerance bounds for computer model predictions in complex situations can require considerable additional methodological development. Two such extensions of the methodology – to functional data – are considered in Bayarri et al. (2005a) and Bayarri et al. (2005b). These extensions also consider uncertainty in the computer model inputs, and could also be utilized for stochastic inputs.

A related approach to Bayesian analysis of computer models is that of Craig et al. (1997), Craig et al. (2001), Goldstein and Rougier (2003) and Goldstein and Rougier (2004), which focus on utilization of linear Bayes methodology to address the problem. Another significant body of work in the computer modeling area is that addressing the importance and uncertainty of input variables and/or the corresponding output distributions (propagation of error) for example, Saltelli et al. (2000), Oakley and O'Hagan (2002), Oakley (2004), Oakley and O'Hagan (2004). The propagation of error issue appears, of course, in a host of scientific applications, a notable recent one being Stainforth et al. (2005).

Overview

Section 1.2 provides an outline of the framework we adopt for computer model validation. The pedagogical example introduced in Section 1.3 gives a simple context in which to display the methodology and its consequences. The second testbed, also introduced in Section 1.3, is a resistance spot welding model; it is succinctly described there with more details available in Bayarri et al. (2002) and Higdon et al. (2004).

The proposed methodology for model evaluation is presented in Sections 2 to 6 with illustrations on the two test bed models. Many technical details are placed in the appendices to prevent their impeding the flow of the arguments. Our presentation is designed so that both engineers and statisticians can follow the process we advance before technical details, notation, etc. are introduced in Section 4 and Section 5. The reader who wishes to jump directly to the specifics of the methodology may go from Section 1.2 to Section 5 or, better still, from Section 2.2 to Section 5.

1.2 Sketch of the framework

Validation can be thought of as a series of activities or steps. These are roughly ordered by the sequence in which they are typically performed. The completion of some or all in the series of activities will often lead to new issues and questions, requiring revision and revisiting of some or all of the activities, even if the model is unchanged. New demands placed on the model and changes in the model through new development make validation a continuing process. The framework must allow for such dynamics.

Step 1. Specify model inputs and parameters with associated uncertainties or ranges – the Input/Uncertainty (I/U) map This step requires considerable subject-matter expertise to help set priorities among a (possibly) vast number of inputs as well as specify uncertainties, often as prior distributions of inputs. As information is acquired through undertaking further steps of the validation process, the I/U map is revisited, revised and updated.

Step 2. Determine evaluation criteria The defining criteria must account for the context in which the model is used, the feasibility of acquiring adequate computer-run and field data, and the methodology to permit an evaluation. In turn the data collection and analyses will be critically affected by the criteria. Moreover, initially stated criteria will typically be revisited in light of constraints and results from later analyses.

Step 3. Data collection and design of experiments Both computer and field experiments are part of the validation (and development) processes; multiple stages of experimentation will be common. The need to design the computer runs along with field experiments can pose non-standard issues. As noted above, any stage of design must interact with other parts of the framework, especially the evaluation criteria.

Step 4. Approximation of computer model output Model approximations (fast surrogates) are usually key for enabling the analyses carried out in Step 5.

Step 5. Analyses of model output; comparing computer model output with field data Uncertainty in model inputs will propagate to uncertainty in model output and estimating the resulting output distribution is often required. The related 'sensitivity analysis' focuses on ascertaining which inputs most strongly affect outputs, a key tool in refining the I/U map.

Comparing model output with field data has several aspects.

- The relation of reality to the computer model ("reality = model + bias"),
- Statistical modelling of the data (computer runs and field data where "field data = reality + measurement error"),
- Tuning/calibrating model input parameters based on the field data,
- Updating uncertainties in the parameters (given the data),
- Accuracy of prediction given the data.

The Bayesian methods (see Section 5) play a crucial role here.

Step 6. Feedback information into current validation exercise and feed-forward information into future validation activities Feedback refers to use of results from Step 5 to improve aspects of the model, as well as to refine aspects of the validation process. Feed-forward refers to the process of utilizing validations of current models to predict the validity of related future models, for which field data are lacking.

1.3 Testbeds

The testbeds provide context for implementing each step of the framework and also prompt consideration of a variety of issues. Testbed 1 is a synthetic example, introduced for pedagogical reasons to illustrate the methodology in a situation where the truth is known; the example should not be viewed as a full model validation study. Testbed 2 is an application drawn from engineering practice.

Testbed 1 – The Pedagogic Example This example is based on a suggestion by G. McRae [personal communication] and it builds on the kinetics of the chemical reaction $SiH_4 \to Si{+}2H_2$. Letting $y(t)$ denote the concentration of SiH_4 as a function of time,

$$dy(t)/dt = -uy(t), \quad y(0) = y_0,$$

where y_0 is the initial concentration and u is an unknown rate that is specific to this chemical reaction; in our later terminology, u is an unknown calibration parameter. As a consequence,

$$y(t) = y_0 \exp(-ut) \ . \tag{1.1}$$

Imagine that, in actual experiments, a residual concentration of c units is left unreacted, so that the kinetics governing the actual experiment is

$$y(t) = (y_0 - c)\exp(-ut) + c \ . \tag{1.2}$$

In our consideration of this example, we will set $y_0 = 5.0$, assumed known to the analyst, and $c = 1.5$ and $u = u_\star = 1.7$, assumed to be unknown. This, of course, is a great oversimplification of a potentially real example; we only use these set values to display various characteristics of the methodology we use. We will utilize the notational convention that a "$_\star$" subscript to a parameter of a computer model indicates the true unknown value of the parameter for the real process under consideration: here, the true chemical reaction rate. (c does not have a star because it is a part of reality that is not even being modeled.) Thus (1.2) is the real process (with the parameter values as indicated above), while (1.1) is to be viewed as the math model that can provide a computer model-based estimate of the concentration of SiH_4 (for any values of t and u). Note that the discrepancy between reality and the computer model – which we will later call the *bias function* – is $c[1 - \exp(u_\star\, t)]$, a nonlinear function of the inputs.

In addition, assume that experiments on the real process are governed by (1.2) subject to a normal measurement error having mean zero (i.e., the experiment is unbiased) and unknown variance. Details concerning the data are given in Section 3.

Testbed 2 – The Spotweld Example In resistance spot welding, two metal sheets are compressed by water-cooled copper electrodes, under an applied load, L. Figure 1 is a simplified representation of the spot weld process, illustrating some of the essential features for producing a weld. A direct current of magnitude C is supplied to the sheets via the two electrodes to create concentrated and localized heating at the interface where the two sheets have been pressed together by the applied load (the so-called faying surface). The heat produced by the current flow across the faying surface leads to melting and, after cooling, a weld "nugget" is formed.

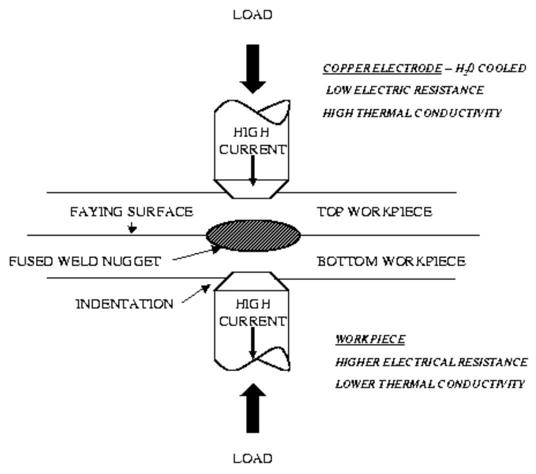


Figure 1: Schematic representation of the spotwelding process.

The resistance offered at the faying surface is particularly critical in determining the magnitude of heat generated. Because contact resistance at the faying surface, as a function of temperature, is poorly understood a nominal function is specified and "tuned" to field data. The effect of this tuning on the behavior of the model is the focus of the example.

The physical properties of the materials will change locally as a consequence of local increase in temperature. Young's modulus and the yield stress of the sheet will fall (that is, the metal will "soften") resulting in more deformation and increase in the size of the faying contact surface, further affecting the formation of the weld. At the same time, the electrical and thermal conductivities will decrease as the temperature rises; all of which will affect the rate of heat generation and removal by conduction away from the faying surface.

The thermal/electrical/mechanical physics of the spot weld process is modelled by a coupling of partial differential equations that govern heat and electrical conduction with those that govern temperature-dependent, elastic/plastic mechanical deformation (Wang and Hayden 1999). Finite element implementations are used to provide a computer model of the electro-thermal conceptual model. Similarly, a finite element implementation is made for the equilibrium and constitutive equations that comprise the conceptual model of mechanical/thermal deformation. These two computer models are implemented using a commercial code (ANSYS).

Key inputs of the model are summarized in Table 1; interesting outputs are discussed in Section 2.2

2 Understanding the Model and Its Uses (Steps 1 and 2)

Understanding the uncertainties associated with the computer model and how the model is used are initial steps in the validation process.

2.1 Step 1. Specify model inputs and parameters with associated uncertainties or ranges – the Input/Uncertainty (I/U) Map

A convenient way to organize information about inputs and their uncertainties is through what we call the Input/Uncertainty map. (This is related to the idea of a PIRT – see Pilch et al. 2001.) The map has four attributes:

a) A list of model features or inputs of potential importance,

b) A ranking of the importance of each input,

c) Uncertainties, either distributions or ranges of possible values, for each input, and

d) Current status of each input describing how the input is currently treated in the model.

The I/U map is dynamic: as information is acquired and the validation process proceeds, the attributes, especially b)-d), may change or require updating.

The inputs are drawn from the development process. They will include parameters inherent to the scientific/engineering assumptions, the mathematical implementation and numerical parameters associated with the implementing code. In short, the inputs are the ingredients necessary to make the model run. Because this list can be enormous, more important parameters must be singled out to help structure the validation process by providing a sense, albeit imperfect, of priorities. We adopt a scale of 1–5 for ranking the inputs with 1 indicating only minor likely impact on prediction error and 5 indicating significant potential impact.

Pedagogic: The two inputs, time t and calibration parameter u, are a complete set of inputs. Each is anticipated to be of rank 5. We are going to focus interest on the first 3.0 units of time, and the range of interesting values of u is going to be $(0, 3)$. A uniform distribution on this range is assumed to capture the uncertainty about the true value of u.

Spotweld: The purpose of the spot weld model is to investigate the process parameters for welding aluminum. The I/U map of the model is in Table 1. The list of inputs in Table 1 is more fully described in Bayarri et al. (2002). Initially, only three inputs have rank 5 based on the model developer's assessment. These three parameters (and gauge) are the focus of the validation experiments; earlier experiments by the model developer led to the impact assessments appearing in the table. The specified ranges of the controllable parameters, current, load, and gauge, are given in Step 2. There is assumed to be "no uncertainty" about these inputs, either in the computer model or in the laboratory data collected for the Validation exercise. (In contrast, if validation of the model were required at the production level, then uncertainties in current and load might be significant–the I/U map is context-dependent.)

There are several specific items connected with the I/U map in Table 1 that are worth noting. First, the most significant specified uncertainty (impact factor 5) in the model is that of contact resistance. The model incorporates contact resistance through an equation that, for the faying surface, has a multiplicative constant u about which it is only known that u lies in the interval

$[0.8, 8.0]$. It will be necessary to *tune* this parameter of the model with field data. The second most significant uncertainty in the model (impact factor 4) is the linear approximation for stress/strain. The modeler was unable to specify the uncertainty regarding this input, and so error in this input will simply enter into the overall unknown (and to be estimated) bias of the model.

INPUT		IMPACT	UNCERTAINTY	CURRENT STATUS
Geometry	electrode symmetry-2d	3	unspecified	fixed
	cooling channel	1	unspecified	fixed
	gauge	unclear	unspecified	1, 2mm
materials		unclear	Aluminum (2 types $\times$ 2 surfaces)	fixed
Stress/ strain	piecewise linear	4	unspecified (worse at high T)	fixed
	C_0, C_1, σ_s	3	unspecified	fixed
contact resistance	$1/\sigma = u \cdot f$; f fixed	3	unspecified	fixed by modeler
	$u = 0$ for electrode/sheet u = tuning for faying	5	$u \in [0.8, 8.0]$	tuned to data for 1 metal
thermal conductivity κ		2	unspecified	fixed
current		5	no uncertainty	controllable
load		5	no uncertainty	controllable
mass density (ρ)		1	unspecified	fixed
specific heat (c)		1	unspecified	fixed
numerical parameters	mesh	1	unspecified	convergence/speed compromise
	M/E coupling time	1	unspecified	
	boundary conditions	1	unspecified	fixed
	initial conditions	1	unspecified	fixed

Table 1: The I/U map for the spot weld model

Initial impact assessments are based on experience to reflect a combined judgment of the inherent sensitivity of the input (the extent to which small changes in the input would affect the output) and the range of uncertainty in the input. These may be revised through sensitivity analyses and 'tuning with data' that occur later in the process. Inputs about which we are "clueless" might be singled out for attention at some point along the validation path but the effect of "missing" inputs (i.e., non-modeled features) may never be quantifiable or only emerge after all effects of "present" inputs are accounted for.

In model validation, attention may need to be paid to the numerical accuracy of the implemented model: for instance, in assessing if numerical solvers and finite element (FEM) codes have 'converged' to the solution of the driving differential equations. This can be important and, as detailed in Cafeo and Cavendish (2001), is an issue of model and code verification. Ideally, numerical accuracy should be addressed early in the model development process and prior to the validation activity emphasized in this paper. It is often the case, however, that convergence will not have been obtained. For example, modelers may simply use the finest mesh size that is computationally feasible, even if insufficient for assuring convergence. The method we propose for validation still works: the error introduced by a lack of convergence becomes part of the 'bias' of the model that is to be assessed (see Section 5). The I/U map should, of course, clearly indicate the situation involving such convergence. The possible confounding effect of parameters, such as grid size, on other assumptions about the model will make it more difficult to improve the model. Ideally, identifying this effect could be done through designed experiments, varying values of the numerical parameters in order to assess numerical accuracy.

2.2 Step 2. Determine evaluation criteria

Evaluation of a model depends on the context in which it is used. Two key elements of evaluation are:

- Specification of an evaluation criterion (or criteria) defined on model output
- Specification of the domain of input variables over which evaluation is sought.

Even if only one evaluation criterion is initially considered, other evaluation criteria inevitably emerge during the validation process. The overall performance of the model may then depend on the outcomes of the validation process for several evaluation criteria – the model may fail for some and pass for others – leading ultimately to follow-on analyses about when and how the model should be used in prediction.

Informal evaluations are typical during the development process – does the computer model produce results that appear consistent with scientific and engineering intuition? Later in the validation process these informal evaluations may need to be quantified and incorporated in the "formal" process. Sensitivity analyses may, in some respects, be considered part of evaluation if, for example, the sensitivities confirm (or conflict with) scientific judgment.

The evaluation criteria can introduce complexities that would need to be addressed at Steps 4–6, but these complexities may also affect the choices made of the criteria. For example, an evaluation criterion that leads to comparisons of curves or surfaces or images places greater demands on the analyst than simpler scalar comparisons.

Of necessity, the specifications must take into account the feasibility of collecting data, particularly field data, to carry out the validation. This can be further complicated by the need to calibrate or tune the model using the collected data; the tuning itself being driven by the specifications.

For the pedagogic example, we are going to be fairly vague about evaluation criteria. We are interested in determining how closely the computer model reproduces the concentration curves we are going to observe in the "field". For spotweld, matters are somewhat more complicated.

Spotweld: Two evaluation criteria were initially posed:

I. Size of the nugget after 8-cycles,

II. Size of the nugget as a function of the number of cycles.

The first criterion is of interest because of the primary production use of the model; the second as a possible aid in reducing the number of cycles to achieve a desired nugget size. Ideally the evaluation would be based directly on the strength of the weld, but weld diameter is taken as a surrogate because of the feasibility of collecting laboratory data on the latter. (Of course, if nugget size was not strongly correlated with weld strength, these criteria would probably be inappropriate.) In production, the spot welding process results in a multiple set of welds, but the evaluation criterion considered here involves only a single weld. Criterion (II) was later discarded as a result of the difficulty during data collection of getting reliable computer runs producing output at earlier times than 8-cycles.

The feasible domains of the input variables were specified to be:

- Material: Aluminum 5182-O and Aluminum 6111-T4,
- Surface: treated or untreated,
- Gauge (mm): 1 or 2,
- Current (kA): 21 to 26 for 1mm aluminum; 24 to 29 for 2mm aluminum,
- Load (kN): 4.0 to 5.3.

Material and surface might enter the model through other input variables relating to properties of materials. Our initial specification in Table 1 considers material and surface as fixed. The tuning parameter, u, has the range indicated In Table 1 and is the only other input that is not fixed.

3 Data Collection (Step 3)

Both computer and field (laboratory or production) experiments are part of the validation and development processes and produce data essential for

- Developing needed approximations to (expensive) numerical models,
- Assessing bias and uncertainty in model predictions,
- Studying sensitivity of a model to inputs,
- Identifying suspect components of models,
- Designing and collecting data that build on, and augment, existing, or historical, data.

The iterative and interactive nature of the validation and development processes will result in multiple stages of computer experiments and even field experiments.

Intuitively, designs should cover the ranges of the key input values and "space-filling" strategies can be devised to accomplish this in an effective way (Sacks et al. 1989; Bates et al. 1996). The specific strategy we use is to select a (maximin) Latin Hypercube Design satisfying $\max_{\text{LHD}} \min_{i,j} \delta(z_i, z_j)$ where δ is a distance on the experimental space. We use code from W. Welch to produce such designs.

t	u	$y^M(\cdot)$
2.159	1.145	0.422
0.941	2.000	0.761
0.303	0.710	4.032
0.709	1.040	2.392
1.753	1.895	0.180
1.144	0.605	2.502
0.506	1.685	2.132
2.391	1.565	0.118
1.956	0.500	1.880
1.550	0.935	1.174
2.594	0.815	0.604
2.797	1.790	0.033
1.347	1.460	0.700
0.100	1.355	4.366
3.000	1.250	0.118

Table 2: Pedagogic example: model data at the 15 selected design points.

Pedagogic: The computer model is given by equation (1.1) (with $y_0 = 5.0$) which is obviously easy to compute. In such cases one can by-pass construction of the computer experiment and development of the model approximation in Section 4, directly incorporating the formula for the computer model into the analyses in Section 5 (see, for example, Paulo et al. (2004)). For illustrative and comparative purposes, however, we treat this problem as if the computer model were in fact computationally intensive to run.

We exercised the computer model (1.1) on a 15-point maximin LHD on the 2-dimensional rectangle $[0.5, 2.0] \times [0.1, 3.0]$ in (u, t) space. The 15 inputs, and corresponding model (function) values are given in Table 2. These will be used in developing the 'fast approximation' to the code.

Spotweld: The inputs to be varied were C = current, L = load, G = gauge, and the unknown tuning parameter u; the other inputs were held fixed. The cost – thirty minutes per computer run – is high, so a limited number, 26, of runs were planned for each of the two gauge sizes. The 26 runs for 1 mm metal covered the 3-dimensional rectangle, $[20, 27] \times [3.8, 5.5] \times [1] \times [1.0, 7.0]$, in the (C, L, G, u) space, while those for the $2mm$ metal covered the 3-dimensional rectangle, $[23, 30] \times [3.8, 5.5] \times [2] \times [0.8, 8.0]$. The explicit values of the 26-point maximin LHDs are given in Table 3, along with the resulting model output for the nugget diameter.

The computer runs exhibited some aberrant behavior. Many (17) runs failed to produce a meaningful outcome at cycle 8; these runs were eliminated. For reasons that are not yet clear many runs were unable to produce reliable data for earlier cycle times; as a result evaluation criteria depending on early cycle times were abandoned. The data retained (35 runs) are used in the subsequent analyses.

Field data will usually be harder to obtain than computer experimental data and, as in spotweld, are often a result of other experiments not designed for the validation study. Typically, field data

Gauge	u	Load	Current	Nugget Dia.		Gauge	u	Load	Current	Nugget Dia.
(mm)	(-)	(kN)	(kA)	(mm)		(mm)	(-)	(kN)	(kA)	(mm)
1	6.52	4.072	26.44	–		2	4.544	3.936	27.76	7.15
1	4.60	4.684	21.68	5.64		2	5.696	4.14	25.52	6.39
1	3.64	5.024	23.64	–		2	1.088	4.684	28.32	6.38
1	7.00	4.412	23.36	–		2	0.8	4.276	24.40	4.87
1	6.76	4.888	25.04	–		2	3.68	4.412	26.08	6.47
1	1.00	4.82	22.52	4.36		2	4.832	4.616	23.00	6.68
1	3.40	4.616	27.00	–		2	7.136	4.344	27.20	6.71
1	5.32	4.48	20.84	6.12		2	4.256	5.228	24.68	6.54
1	2.92	5.092	20.56	5.00		2	3.392	4.004	23.28	5.97
1	1.48	5.364	21.12	4.53		2	1.952	4.48	23.84	5.72
1	2.20	4.004	21.40	5.20		2	2.528	3.8	24.96	6.23
1	2.68	4.344	25.88	–		2	2.24	4.208	29.72	–
1	2.44	5.50	23.08	–		2	1.376	5.024	25.80	5.46
1	4.36	3.80	25.32	–		2	7.424	4.072	28.88	–
1	1.24	4.208	24.76	6.06		2	6.272	4.548	29.16	7.36
1	6.04	4.752	20.00	–		2	6.848	5.364	23.56	–
1	5.56	5.432	25.60	–		2	3.968	4.888	29.44	7.16
1	1.96	4.956	26.16	6.69		2	3.104	5.432	28.60	6.61
1	5.80	3.936	23.92	7.17		2	5.12	5.5	26.64	5.98
1	4.84	4.14	22.80	–		2	6.56	3.868	26.36	6.74
1	3.16	3.868	22.24	5.71		2	5.984	4.956	24.12	5.32
1	6.28	5.228	21.96	5.38		2	8	5.092	28.04	–
1	1.72	4.548	24.20	5.85		2	2.816	4.82	26.92	6.70
1	5.08	5.16	26.72	–		2	5.408	5.16	30.00	–
1	4.12	5.296	24.48	6.87		2	1.664	5.296	27.48	6.02
1	3.88	4.276	20.28	4.91		2	7.712	4.752	25.24	5.50

Table 3: Spotweld data from 52 model runs. Run failures indicated by –

will depend crucially on the specifications in Section 2.2 and what can be feasibly obtained; specific design strategies usually seem to have little role.

The field data obtained in the two test beds is as follows.

Pedagogic: In order to simulate field data, recall that "reality" was specified by (1.2) with $c = 1.5$ and $u_\star = 1.7$. For these values, we computed equation (1.2) at an equally-spaced grid of ten values of t in the interval $(0.11, 3.01)$. For each value of these ten true values, we obtained 3 replicate field observations by adding independent $N(0, 0.3^2)$ noise to the true value. The resulting data can therefore be thought of as the result of three independent replicates in a lab experiment measuring the concentration of SiH_4 on the specified grid of time points. These data are given in Table 4. In the analysis, we will, of course, presume that $u_\star$ and c are unknown.

Spotweld: The field data for spotweld is given in Table 5. It was obtained by physical experimentation, the details of which made reasonable the assumption that the measurement errors are independent normal, with mean zero and unknown variance.

t	$y^F(\cdot)$		
0.110	4.730	4.720	4.234
0.432	3.177	2.966	3.653
0.754	1.970	2.267	2.084
1.077	2.079	2.409	2.371
1.399	1.908	1.665	1.685
1.721	1.773	1.603	1.922
2.043	1.370	1.661	1.757
2.366	1.868	1.505	1.638
2.688	1.390	1.275	1.679
3.010	1.461	1.157	1.530

Table 4: Pedagogic example: field data consists of 3 replicate observations of the process (plus noise) at each of 10 input values.

L	C	G	$y^F(\cdot)$									
4.00	21.0	1	4.81	5.08	5.09	4.84	5.40	5.14	4.92	5.31	4.95	4.80
4.00	23.5	1	5.31	6.52	5.89	5.51	5.77	4.96	5.04	5.22	5.54	6.36
4.00	26.0	1	5.52	6.62	5.97	5.76	6.13	5.82	5.81	6.00	6.00	6.52
5.30	21.0	1	5.09	4.43	4.63	5.01	5.07	4.14	4.03	4.30	4.09	4.02
5.30	23.5	1	5.11	5.17	5.71	5.60	5.85	4.60	5.51	4.82	6.37	5.23
5.30	26.0	1	5.34	5.19	5.86	5.94	5.98	5.09	5.43	5.14	5.21	5.73
4.00	24.0	2	6.78	5.89	6.49	6.78	6.81	7.00	7.16	6.68	6.68	6.98
4.00	26.5	2	6.62	6.54	6.30	6.00	6.67	6.89	7.15	5.99	5.90	7.29
4.00	29.0	2	7.28	6.98	7.46	7.87	8.02	6.97	8.15	7.14	7.55	7.75
5.30	24.0	2	6.62	6.74	6.59	6.39	6.45	6.64	5.59	6.30	5.64	6.05
5.30	26.5	2	7.25	6.80	6.50	6.36	7.67	7.14	5.95	7.10	7.57	7.08
5.30	29.0	2	7.62	7.71	8.14	7.26	8.37	7.68	6.95	6.41	8.35	7.50

Table 5: Spotweld example: field data consists of 10 replicate observations of nugget size at each of 12 input values.

Note: In both examples, replicated data was available at the various input values. Having such replicate data is highly desirable, in that doing a reasonable job of pinning down the measurement error variance makes the validation analysis considerably more accurate.

4 Model Approximation (Step 4)

4.1 Introduction

Unless the computer model code is very cheap to run, it is difficult to use the code directly to perform the validation analysis, since validation (see Section 5) typically requires many code evaluations. It is thus common to utilize approximations to the computer model – based on a limited number of runs – for validation. There are other reasons for desiring such approximations, such as ease of use "in the field" (as compared to use of the original code), in optimization (where typical algorithms may again require many evaluations of the code), and in 'output analysis' (which is analysis of senstivity of outputs to inputs or analysis of output distributions based on random inputs.)

A very useful general tool for models whose output depends smoothly on inputs (very common in engineering and scientific processes) is the Gaussian process response surface technique (GASP) advanced in Sacks et al. (1989) and frequently utilized subsequently (Currin et al. 1991; Morris et al. 1993; Kennedy and O'Hagan 2001; Santner et al. 2003; Higdon et al. 2004). This technique meshes well with the validation analysis proposed in Step 5.

More formally, denote model output by $y^M(\boldsymbol{x}, \boldsymbol{u})$, where $\boldsymbol{x}$ is a vector of controllable inputs and $\boldsymbol{u}$ is a vector of unknown calibration and/or tuning parameters in the model. The goal is to approximate $y^M(\boldsymbol{x}, \boldsymbol{u})$ by a function $\hat{y}^M(\boldsymbol{x}, \boldsymbol{u})$ that is easy to compute. In addition, it is desirable to have a variance function $V^M(\boldsymbol{x},\boldsymbol{u})$ that measures the accuracy of $\hat{y}^M(\boldsymbol{x}, \boldsymbol{u})$. We turn now to the details of how the GASP approach achieves these goals.

4.2 The GASP response-surface methodology

Let $\boldsymbol{y}^M = (y^M(\boldsymbol{x}_1, \boldsymbol{u}_1), \ldots, y^M(\boldsymbol{x}_m, \boldsymbol{u}_m))$ denote the vector of m evaluations of the model at inputs $D^M = \{(\boldsymbol{x}_i, \boldsymbol{u}_i) : i = 1, \ldots, m\}$ and write $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{u})$. The computer model is exercised only at the inputs D^M, so that $y^M(\boldsymbol{z})$ is effectively unknown for other inputs $\boldsymbol{z} \notin D^M$. Before seeing $\boldsymbol{y}^M$, we assign $y^M(\cdot)$ a prior distribution, specifically, a stationary Gaussian process with mean and covariance functions governed by unknown parameters $\boldsymbol{\theta}^L$ and $\boldsymbol{\theta}^M = (\lambda^M, \boldsymbol{\alpha}^M, \boldsymbol{\beta}^M)$, respectively. (Since in applications we only deal with a finite set of $\boldsymbol{z}_i$, the Gaussian process at these points reduces to a multivariate normal distribution, so in essence we are assuming that the output of the code at any finite number of locations has a multivariate normal distribution.)

The mean function of the Gaussian process is assumed to be of the form $\boldsymbol{\Psi}'(\cdot)\boldsymbol{\theta}^L$ where $\boldsymbol{\Psi}(\boldsymbol{z})$ is a *specified* $k \times 1$ vector function of the input $\boldsymbol{z}$ and $\boldsymbol{\theta}^L$ is a $k \times 1$ vector of unknown parameters. A constant mean ($k = 1$, $\Psi(\boldsymbol{z}) = 1$, and $\boldsymbol{\theta}^L = \theta$) is often satisfactory if one plans only to use the model approximation within the range of the available model-run data. A more complicated mean function can be useful if the model approximation is to be used outside the range of the data because outside of this range the Gaussian process approximation to the model will gradually tend towards its estimated mean function. This can be especially important when features such as temporal trends are present.

The parameter λ^M is the precision (the inverse of the variance) of the Gaussian process and the other parameters $(\boldsymbol{\alpha}^M, \boldsymbol{\beta}^M)$ control the correlation function of the Gaussian process, which we assume to be of the form

$$c^M(\boldsymbol{z}, \boldsymbol{z}^\star) = \exp\left(-\sum_{j=1}^{d} \beta_j^M |z_j - z_j{}^\star|^{\alpha_j^M} \right). \tag{4.3}$$

Here, d is the number of coordinates in $\boldsymbol{z}$, the α_j^M are numbers between 1 and 2, and the β_j^M are positive scale parameters. The product form of the correlation function (each factor is itself a correlation function in one-dimension) helps the computations made later. Prior beliefs about the smoothness properties of the function will affect the choice of $\boldsymbol{\alpha}^M$. The choice $\alpha_j^M = 2$ for all j reflects the belief that the function is infinitely differentiable, which is plausible for many engineering and scientific models.

This can be summarized by saying that, given the hyper-parameters $\boldsymbol{\theta}^L$ and $\boldsymbol{\theta}^M = (\lambda^M, \boldsymbol{\alpha}^M, \boldsymbol{\beta}^M)$, the prior distribution of y^M is $\mathrm{GP}(\boldsymbol{\Psi}'(\cdot)\,\boldsymbol{\theta}^L, \frac{1}{\lambda^M}\, c^M(\cdot,\cdot)\,)$, i.e., a Gaussian process with the given mean and covariance functions.

As before, let $\boldsymbol{y}^M$ denote the vector of model evaluations at the set of inputs D^M. Conditionally on the hyperpartameters, $\boldsymbol{y}^M$ is, *a priori*, multivariate normal with covariance matrix $\boldsymbol{\Gamma}^M = \boldsymbol{C}^M(D^M, D^M)/\lambda^M$, where $\boldsymbol{C}^M(D^M, D^M)$ is the matrix with (i,j) entry $c^M(\boldsymbol{z}_i, \boldsymbol{z}_j)$, for $\boldsymbol{z}_i, \boldsymbol{z}_j$ in D^M.

After observing $\boldsymbol{y}^M$, the conditional posterior distribution of y^M given the hyperparameters, $p(y^M(\cdot) \mid \boldsymbol{y}^M, \boldsymbol{\theta}^L, \boldsymbol{\theta}^M)$, is a Gaussian process with updated mean and covariance functions given by

$$\mathrm{E}[\, y^M(\boldsymbol{z}) \mid \boldsymbol{y}^M, \boldsymbol{\theta}^L, \boldsymbol{\theta}^M] = \boldsymbol{\Psi}'(\boldsymbol{z})\,\boldsymbol{\theta}^L + \boldsymbol{r}_z{}'(\boldsymbol{\Gamma}^M)^{-1}(\boldsymbol{y}^M - \boldsymbol{X}\boldsymbol{\theta}^L) \tag{4.4}$$

$$\mathrm{Cov}[\, y^M(\boldsymbol{z}), y^M(\boldsymbol{z}^\star) \mid \boldsymbol{y}^M, \boldsymbol{\theta}^L, \boldsymbol{\theta}^M] = \frac{1}{\lambda^M}\, c^M(\boldsymbol{z}, \boldsymbol{z}^\star) - \boldsymbol{r}_z{}'(\boldsymbol{\Gamma}^M)^{-1}\boldsymbol{r}_{z^\star}, \tag{4.5}$$

where $\boldsymbol{r}_z{}' = \frac{1}{\lambda^M}\,(c^M(\boldsymbol{z}, \boldsymbol{z}_1), \ldots, c^M(\boldsymbol{z}, \boldsymbol{z}_m))$, $\boldsymbol{\Gamma}^M$ is given above and $\boldsymbol{X}$ is the matrix with rows $\boldsymbol{\Psi}'(\boldsymbol{z}_1), \ldots, \boldsymbol{\Psi}'(\boldsymbol{z}_m)$.

With specifications for $\boldsymbol{\theta}^L$ and $\boldsymbol{\theta}^M$, the GASP behaves as a Kalman filter, yielding a posterior mean function (4.4) that can be used as the fast approximation or inexpensive emulator for $y^M(\cdot)$. Thus (given $(\boldsymbol{\theta}^L, \boldsymbol{\theta}^M)$), the response surface approximation to $y^M(\boldsymbol{z})$, at any point $\boldsymbol{z}$, is simply $\mathrm{E}[\, y^M(\boldsymbol{z}) \mid \boldsymbol{y}^M, \boldsymbol{\theta}^L, \boldsymbol{\theta}^M]$ given by (4.4), and the variance measuring the uncertainty in this approximation is, following (4.5), $\mathrm{Var}[\, y^M(\boldsymbol{z}) \mid \boldsymbol{y}^M, \boldsymbol{\theta}^L, \boldsymbol{\theta}^M] = 1/\lambda^M - \boldsymbol{r}_{\boldsymbol{z}}'(\boldsymbol{\Gamma}^M)^{-1}\boldsymbol{r}_{\boldsymbol{z}}$. Note that the variance is zero at the design points at which the function was actually evaluated: the GASP approximation is an interpolator of the data.

Unfortunately, the hyper-parameters $(\boldsymbol{\theta}^L, \boldsymbol{\theta}^M)$ are rarely, if ever, known. Two possibilities then arise:

1. Plug-in some estimates in the above formulae, for instance maximum likelihood estimates (as in the GASP software of W. Welch), pretending they are the 'true' values. For MLE estimates $(\hat{\boldsymbol{\theta}}^L, \hat{\boldsymbol{\theta}}^M)$, this produces the following model approximation for input $\boldsymbol{z}$:

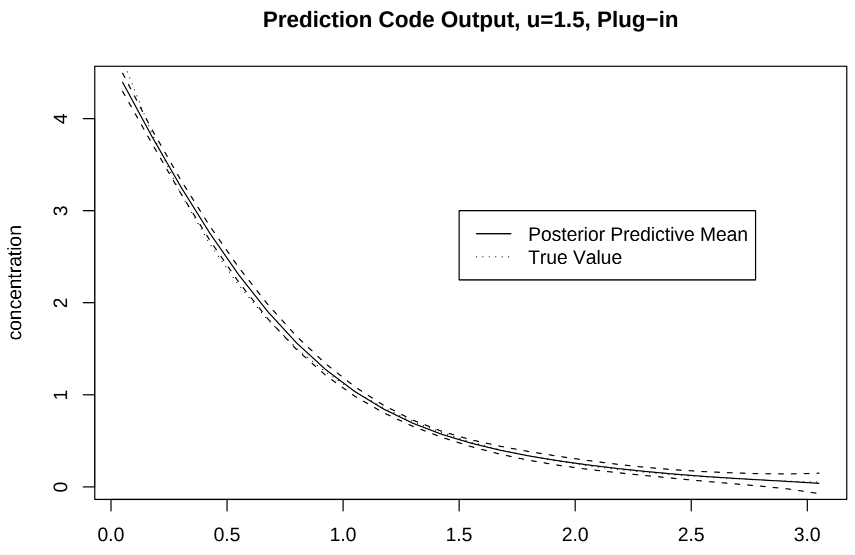
$$\hat{y}^{\text{MLE}}(\boldsymbol{z}) = \boldsymbol{\Psi}'(\boldsymbol{z})\,\hat{\boldsymbol{\theta}}^L + \hat{\boldsymbol{r}}'_{\boldsymbol{z}}(\hat{\boldsymbol{\Gamma}}^M)^{-1}(\boldsymbol{y}^M - \boldsymbol{X}\hat{\boldsymbol{\theta}}^L)\,,$$

where $\hat{\boldsymbol{\theta}}^M = (\hat{\lambda}^M, \hat{\boldsymbol{\alpha}}^M, \hat{\boldsymbol{\beta}}^M)$ is used to compute $\hat{\boldsymbol{\Gamma}}^M$ and $\hat{\boldsymbol{r}}_{\boldsymbol{z}}$. Similarly, $\text{Var}^{\text{MLE}}[y^M(\boldsymbol{z}) \mid \boldsymbol{y}^M, \hat{\boldsymbol{\theta}}^L, \hat{\boldsymbol{\theta}}^M] = 1/\hat{\lambda}^M - \hat{\boldsymbol{r}}'_{\boldsymbol{z}}(\hat{\boldsymbol{\Gamma}}^M)^{-1}\hat{\boldsymbol{r}}_{\boldsymbol{z}}$ is used as the estimate of the approximation variance. This results in an underestimate of the true variability, since the uncertainty in the estimates of $\hat{\boldsymbol{\theta}}^L$ and $\hat{\boldsymbol{\theta}}^M$ is not taken into account.

2. Integrate the hyper-parameters with respect to the posterior distribution in a full Bayesian analysis (as detailed in Paulo 2005), leading to a more appropriate approximation: the integral of (4.4) with respect to the posterior distribution of $(\boldsymbol{\theta}^L, \boldsymbol{\theta}^M)$, $p(\boldsymbol{\theta}^L, \boldsymbol{\theta}^M \mid \boldsymbol{y}^M)$. This is done in practice by, using MCMC techniques, generating N (large) values $\{(\boldsymbol{\theta}^{L(i)}, \boldsymbol{\theta}^{M(i)})\}$ from this posterior distribution, evaluating (4.4) at these generated values, and averaging. The variance of this approximation is obtained by adding two terms: the posterior expectation of (4.5) and the posterior variance of (4.4). In practice, these terms are estimated, respectively, by the sample average of (4.5) and by the sample variance of (4.4) evaluated at the generated values $(\boldsymbol{\theta}^{L(i)}, \boldsymbol{\theta}^{M(i)})$. Alternatively, one may wish to draw realizations from the marginal posterior of $y^M(\boldsymbol{z})$, $p(y^M(\boldsymbol{z}) \mid \boldsymbol{y}^M)$, directly, and then compute appropriate summary statistics. This can be done in practice by, for each generated value $(\boldsymbol{\theta}^{L(i)}, \boldsymbol{\theta}^{M(i)})$ computing (4.4) and (4.5) and then drawing a normal random variable with mean and variance given by these numbers.

Pedagogic: In the setting of the pedagogic example, t is a controllable input and u is a calibration parameter. Figure 2 shows the approximations and associated 90% pointwise posterior intervals for the output of model (1.1), as a function of t, at the input value $u = 1.5$. The top panel was produced using the MLE plug-in strategy, while the bottom panel is the result of a full Bayesian analysis. These approximations are based on applying the GASP methodology to the the 15 model evaluations given in Table 2. Note that the approximations track the true model (function) values very well. Note, also, that the approximation has nearly zero variance when it is being done near one of the input values at which the computer model was evaluated.

Spotweld: The vector of controllable inputs is $\boldsymbol{x} = (C, L, G)$, the tuning parameter is u. Use of a GASP full Bayesian analysis with the data from Table 3 leads to the response surface approximation to $y^M(C, L, G, u)$ that can be seen in Figure 8 of Higdon et al. (2004). The MLE approximation is very similar, and hence is omitted.



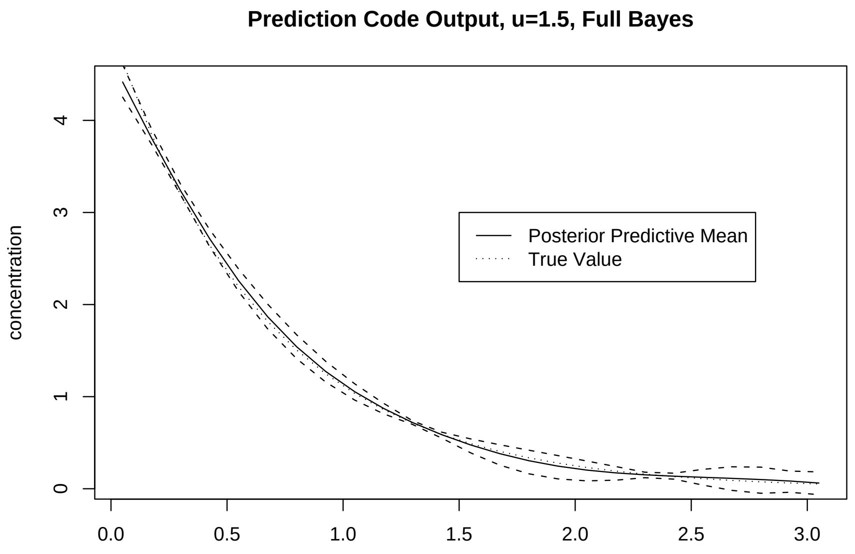


Figure 2: Pedagogic Example: Prediction of $y^M(\cdot, u)$ where $u = 1.5$ using GASP approximation. Top panel corresponds to the maximum likelihood plug-in approach, bottom panel corresponds to full Bayesian approach. Dashed lines are pointwise 90% confidence bands.

4.3 MLE plug-in or full Bayes?

The full Bayesian analysis is theoretically superior, because the resulting variance takes into account the uncertainty in the GASP parameters. Thus, in Figure 2, the full Bayes analysis does indicate considerably larger uncertainty for certain values of t. Because the function being approximated is very smooth in this example, the additional uncertainty is not really needed, but it could be relevant if approximating less smooth models. The advantage of using the MLE plug-in approach is, of course, computational; it is much easier to implement a GASP with fixed parameters than averaging GASPs over a posterior sample of parameters.

The primary focus in this paper is not in model approximation itself, but in the validation/prediction analysis discussed in the next section. In such analyses, we have found that use of the maximum likelihood estimates of the GASP parameters typically yields much the same answers as the full Bayesian analysis, at least when tuning/calibration parameters are present in the computer model. The reason is that the uncertainty in calibration and tuning parameters, together with the uncertainty in the 'bias' of the computer model, tend to overwhelm the uncertainty in the model approximation. Hence our current (cautious) recommendation is to use MLE plug-in GASPs, together with Bayesian analysis of the validation/prediction process. This allows implementation of the validation methodology in vastly more complicated scenarios (Bayarri et al. 2005a) than would otherwise be possible.

5 Analysis of Model Output (Step 5)

In this section, we describe the structure (statistical model) and analysis we use for computer model evaluation, and illustrate the methods using the test bed examples. Some technical details that threaten to cloud the exposition are relegated to appendices.

The section is organized as follows: we start by describing the statistical structure and necessary notation. In Section 5.2 we address computation of the posterior distributions, predictions and tolerance bounds, the heart of the matters at hand.

5.1 Notation and statistical modeling

The computer model approximates reality and the discrepancy between the model and reality is the model bias. Accounting for this bias is the central issue for validation. There are (at least) three sources for this bias:

1. The science or engineering used to construct the model is incomplete.
2. Calibrated/tuned parameters may be in error.
3. Numerical implementation may introduce errors (e.g., may not have converged).

The first two sources are typical; the third occurs with some frequency.

The computer model alone cannot provide evidence of bias. Either expert opinion or field data are necessary to assess bias – we focus on the latter. If field data are unavailable (even from experiments involving related models), strict model validation is impossible. Useful things may still be said, but the ultimate goal of being able to confirm accuracy of predictions will not be attainable.

Recall that $y^M(\boldsymbol{x}, \boldsymbol{u})$ denotes the model output when $(\boldsymbol{x}, \boldsymbol{u})$ is input. When $\boldsymbol{u}$ is not present, we formalize the statement "reality = model + bias" as

$$y^R(\boldsymbol{x}) = y^M(\boldsymbol{x}) + b(\boldsymbol{x}) \,, \tag{5.6}$$

where $y^R(\boldsymbol{x})$ is the value of the 'real' process at input $\boldsymbol{x}$ and $b(\boldsymbol{x})$ is the (unknown) bias function. When $\boldsymbol{u}$ is present as a calibration parameter, we call its true (but unknown) value $\boldsymbol{u}_\star$, and then bias is defined via

$$y^R(\boldsymbol{x}) = y^M(\boldsymbol{x}, \boldsymbol{u}_\star) + b_{\boldsymbol{u}_\star}(\boldsymbol{x}) \ . \tag{5.7}$$

In situations where $\boldsymbol{u}$ is viewed as simply a tuning parameter there is no 'true value', so $\boldsymbol{u}_\star$ should be thought of as some type of best fit value of $\boldsymbol{u}$, with the bias defined relative to this. Note that there is confounding between $\boldsymbol{u}_\star$ and the bias function, i.e. they are not statistically identifiable. This important issue is discussed in Section 5.3, as it has profound implications for the possible types of analysis; in particular the natural way to deal with a lack of identifiability is to utilize prior information to provide identification or, at least, use Bayesian analysis to properly account for the uncertainty caused by the non-identifiability. For notational convenience we will often drop the dependence of b on the true value of the calibration parameter.

Field data at inputs $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ are assumed to be "reality" measured with error. Specifically,

$$y^F(\boldsymbol{x}_i) = y^R(\boldsymbol{x}_i) + \epsilon_i^F \tag{5.8}$$

where the ϵ_i^F are independent normal random errors with mean zero and variance $1/\lambda^F$. This equation may only be reasonable after suitable transformation of the data and, often, more complicated error structures (such as correlated errors) are needed; these can typically be accommodated with some additional computational effort. Note that $\boldsymbol{u}$ is *not* an input in determining the field data.

The assumption that ϵ^F has mean zero implies there is no bias in the field measurements i.e., the measurement process is "well-calibrated". Otherwise, the situation is problematic: the estimated bias will be a combination of both model and field bias, and there is no data-based way to separate the two; additional insight or expert opinion would be necessary to permit such separation. Unfortunately, it is quite common for 'existing field data' (e.g., historical data, data acquired for different purposes but now used for validation) to be biased (see, e.g., Roache 1998), so obtaining unbiased field data may be challenging in its own right (see Trucano et al. 2002, for further discussion).

For the Bayesian model to be complete, one needs to specify the priors for the unknowns: $\boldsymbol{u}$, λ^F and $b(\boldsymbol{x})$. These are chosen as follows:

— $p(\boldsymbol{u})$ is specified in the I/U map; it is often uniform on a given range;

— $p(\lambda^F)$ is Exponential (see Appendix A.9);

— the prior for the bias function will be a GASP (see below and Appendix A.9).

If computation of y^M (as in the pedagogic example) is fast, the Bayesian analysis can proceed directly. Otherwise (as in spotweld where a single model run may take 30 minutes) we need to also incorporate the model approximation from Section 4 into the Bayesian analysis. We must then either add the GASP (hyper-)parameters for y^M to the list of unknowns for a complete Bayesian analysis, or use the plug-in MLE method if required by computational limitations. See Section 5.2.1 below for details.

We choose the GASP for the bias to have correlation function of the same form as in (4.3), with its own set of covariance parameters $(\lambda^b, \boldsymbol{\beta}^b, \boldsymbol{\alpha}^b)$, but with all components of $\boldsymbol{\alpha}^b$ set at 2. Restricting $\boldsymbol{\alpha}^b$ at 2 (or even at some other value such as 1.9) reduces the number of hyper-parameters that must be taken into account. Because the bias cannot be observed directly and field data are usually scant, the information about the hyper-parameters is limited and reducing their number is computationally advantageous. Moreover, predictions and their error bounds will only be marginally affected by imposing this restriction. In fact, the restriction implies that the bias is very smooth, a condition all but certain to hold where reality, y^R, is smooth, a typical state in engineering and scientific applications; this smoothness assumption is also of help in de-confounding the bias and $\boldsymbol{u}$.

The mean function of the GASP for the bias process is typically chosen to be either zero or an unknown constant μ^b. Since the bias is not directly observed, it is doubtful whether more complicated mean structures are viable. For interpolation, the choice between zero mean or unknown level will have marginal effect on the results of the analysis; for extrapolation, however, as in the case of the mean of the GASP approximation to the code output described in Section 4, the latter choice might be more appropriate, as it may affect predictions and associated tolerance bounds (to be precisely defined in Section 5.2.3). Also, allowing an unknown mean level for the bias may reduce the danger of over-tuning, as indicated in Section 5.3. As in the case of the GASP approximation to the computer model, one can also utilize a plug-in method to determine the GASP correlation parameters, if required for computational simplification. This is discussed in Section 5.2.1 below.

5.2 Bayesian inferences

5.2.1 The posterior distribution and its computation

Begin with the setting where the computer code is fast so approximation of y^M is not necessary. The modeling assumptions from Section 5.1 are that, for each field input $\boldsymbol{x}$,

$$\begin{aligned} y^F(\boldsymbol{x}) &= y^R(\boldsymbol{x}) + \epsilon^F \\ y^R(\boldsymbol{x}) &= y^M(\boldsymbol{x}, \boldsymbol{u}) + b_{\boldsymbol{u}}(\boldsymbol{x}) \\ \epsilon^F &\sim N(0, 1/\lambda^F)\,. \end{aligned}$$

Given the unknowns, these produce a multivariate normal density for the collection of all field data, $\boldsymbol{y}^F$, denoted by $f(\boldsymbol{y}^F \mid \boldsymbol{u}, \lambda^F, b)$. (Strictly, we should write $\boldsymbol{u}_\star$ instead of $\boldsymbol{u}$ but, in the Bayesian approach, all unknowns are considered to be random and so we will drop the $\star$ subscript for notational simplicity. Also suppressed is the dependence of b on $\boldsymbol{u}$.) Denote the prior distribution of the unknown elements $(\boldsymbol{u}, \lambda^F, b)$ by $p(\boldsymbol{u}, \lambda^F, b)$. (Prior construction was already described briefly in Section 5.1; details are in Appendix A.9.) Write the posterior density of these unknowns, given the data $\boldsymbol{y}^F$, as

$$p(\boldsymbol{u}, \lambda^F, b \mid \boldsymbol{y}^F) \propto f(\boldsymbol{y}^F \mid \boldsymbol{u}, \lambda^F, b)\; p(\boldsymbol{u}, \lambda^F, b). \tag{5.9}$$

The posterior distribution is determined via MCMC techniques (cf. Robert and Casella 1999). Carrying out the MCMC analysis requires evaluation of $y^M(\boldsymbol{x}, \boldsymbol{u})$ at each generated value of $\boldsymbol{u}$ and $\boldsymbol{x}$ in the field design space D^F. This is infeasible when model runs are expensive, in which case we resort to the GASP approximation of y^M, described in Section 4, to carry out the computations. This (unavoidably) introduces additional uncertainty into the predictions.

Two key simplifications: For reasons that have to do with achieving a stable and quasi-automatic MCMC algorithm, we recommend two simplifications, which together we call a *Modular-MLE* analysis:

1. Utilize a *modular analysis*, in which the the GASP hyperparameters for the computer model are determined only from the computer model data. In a full Bayesian analysis, the field data could also influence these hyperparameters. There are scientific as well as computational reasons for utilizing the modular approach. These, and implementation issues, are discussed in Appendix A.8.

2. Rather than keeping GASP hyperparameters random in the Bayesian analysis, fix them (for both the computer model and the bias) at their MLE's; leave only the precisions and calibration parameters random. (Details on how these estimates are computed are given in Appendix A.9.) The reason for doing this is partly computational, and partly to ensure that the methodology is stable and quasi-automatic. Further discussion of this is given in Appendix A.9.

Despite the fact that the modular-MLE analysis is only approximately Bayes, the resulting answers seem to be close to those from a full Bayesian analysis, at least when it comes to prediction (see Section 5.2.4 below). We note that this type of approximation was also utilized in, e.g., Kennedy and O'Hagan (2001).

The MCMC analysis that results (see Appendix B for details) produces a set of N draws from the posterior distribution of the unknowns $\boldsymbol{u}, \lambda^F$, $y^M(\boldsymbol{x}, \boldsymbol{u})$, and b. To be more precise, the output of the computations is a sample $\{\boldsymbol{u}^{(i)}, \lambda^{F(i)}, y^M(\boldsymbol{x}, \boldsymbol{u}^{(i)}), b^{(i)}(\boldsymbol{x}), i = 1, \ldots, N\}$. From these samples, the posterior distribution of all quantities of interest can be estimated.

As an example, the posterior distributions of calibration or tuning parameters can be estimated by a histogram computed from the samples of the $\boldsymbol{u}^{(i)}$. From these samples one can also form an estimate, $\hat{\boldsymbol{u}}$, of the unknown $\boldsymbol{u}$; for instance, the average of the samples is an approximation to the posterior mean of $\boldsymbol{u}$. Credible intervals for $\boldsymbol{u}$ can be formed by taking appropriate percentiles of the ordered samples.

Pedagogic: The controllable input is $x = t$ and the calibration parameter is u. Recall that the data were generated with $u_\star = 1.7$ and an added bias term. The posterior distribution of u from the MCMC analysis is given in the upper right graph in Figure 3. The estimated posterior mean of u is $\hat{u} = 1.29$ with a 90% credible interval of (0.49,2.87).

Although the true value $u_\star = 1.7$ lies within the credible interval, the posterior distribution does not concentrate near the true value, a consequence of the confounding between calibration and bias. Unless bias is negligible, the posterior distribution of u cannot be relied upon to accurately estimate calibration parameters.

Of course, estimating u while ignoring bias leads to even less reliable results. For instance, those who ignore bias would typically use the least-squares estimate of u here, which is $\tilde{u} = 0.63$ – extremely remote from the true value. The Bayesian analysis yields more accurate values of u because it compromises between bias and model-tuning.

Spotweld: The vector of controllable inputs is $\boldsymbol{x} = (C, L, G)$ and there is a tuning parameter u. Figure 4 gives the posterior density of u based on the Modular-MLE approach. The estimated posterior mean is $\hat{u} = 3.28$. There is clearly considerable uncertainty in values for u. Assessments of prediction accuracy (described in Section 5.2.2) account for this uncertainty and help alleviate the danger of over-tuning that can result if one were to simply pick and use a single fixed parameter value such as 3.28.

The considerable right tail here is likely due to the fact that there was data from two thicknesses (gauges) of material – the 'optimal' tuning parameter for each gauge would be different. This again indicates how misleading it would be to simply choose a best estimate of the tuning parameter, and proceed as if it were known. Note that the full-Bayesian analysis in Higdon et al. (2004) leads to a qualitatively similar posterior.

Similarly, the estimated bias function is given by

$$\hat{b}(\boldsymbol{x}) = \frac{1}{N} \sum_{i=1}^{N} b^{(i)}(\boldsymbol{x}).$$

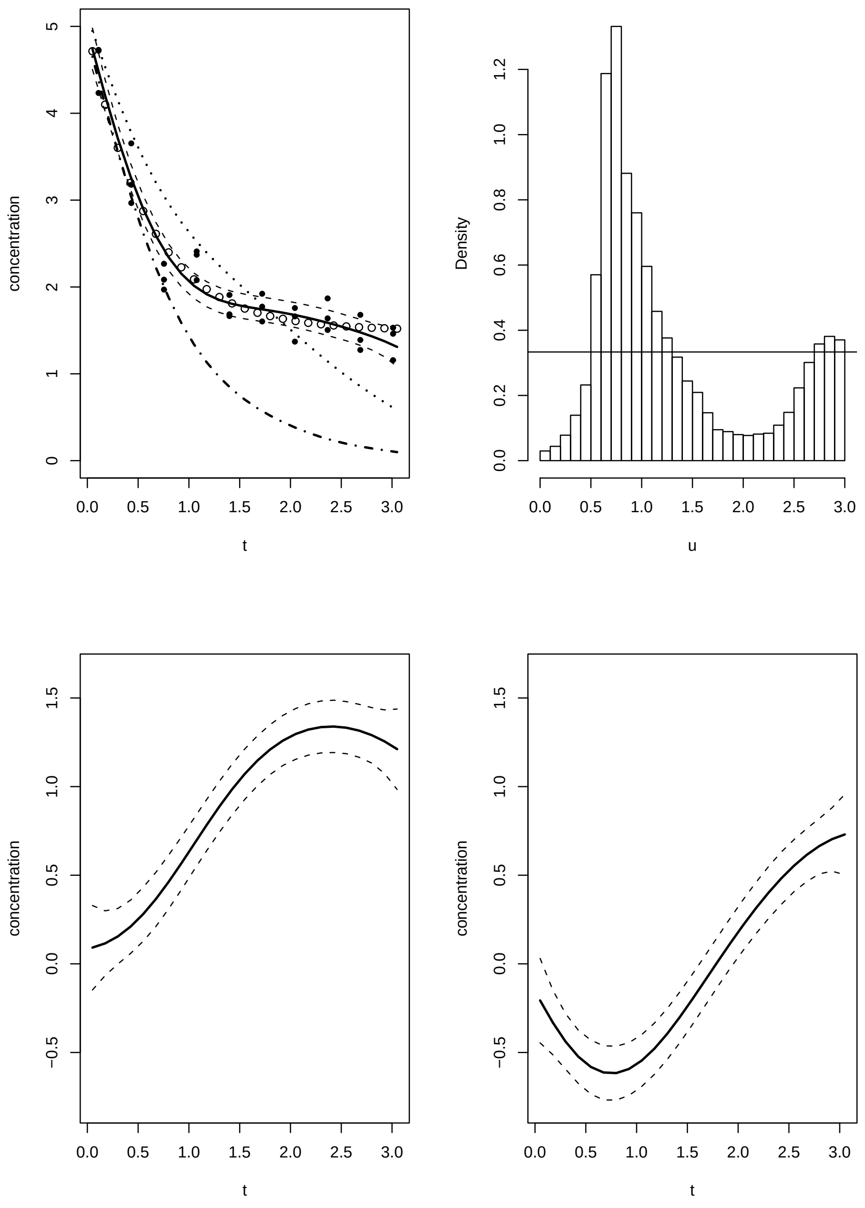


Figure 3: Pedagogic Example: Top-Left Panel: The solid line is the bias-corrected prediction, the dashed lines are 90% tolerance bounds. The dash-dotted and dotted lines correspond , respectively, to the pure-model prediction associated to the posterior mean and to the least-squares estimate of u. The circles are the true values of the real process, whereas the triangles represent the observed field data. Top-Right Panel: Posterior distribution of u. Lower Panels: Estimate of bias corresponding to the pure-model prediction associated to the posterior mean of u (left plot) and to the least-squares estimate of u (right plot).

Separately graphing this bias function is not particularly useful, because of its very considerable posterior dependence on $\boldsymbol{u}$. Thus, when we present bias functions in later figures, we will give them conditionally on interesting values of $\boldsymbol{u}$.

5.2.2 Predictions and bias estimates

The central issue for validation is assessing whether the accuracy of the predictions produced by the computer model is adequate for the intended use of the model. The MCMC samples described above can be used to produce predictions with associated uncertainties, thus quantifying validation.

For instance, to predict the real process $y^R(\boldsymbol{x})$ at a set of (new) inputs D^F_{NEW} (denoting the resulting vector by $\boldsymbol{y}^R_{\text{NEW}}$), all we need is to have access to draws from the posterior predictive distribution of $\boldsymbol{y}^R_{\text{NEW}}$, $p(\boldsymbol{y}^R_{\text{NEW}} \mid \boldsymbol{y}^F, \boldsymbol{y}^M)$, where $\boldsymbol{y}^F$ and $\boldsymbol{y}^M$ are the available field and model data. Because of equation (5.7), these are obtained from draws from the joint posterior predictive of $\boldsymbol{y}^M_{\text{NEW}}$ and $\boldsymbol{b}_{\text{NEW}}$. Denote these draws by

$$\boldsymbol{y}^{M(i)}_{\text{NEW}},\ \boldsymbol{b}^{(i)}_{\text{NEW}},\ i = 1, \ldots, N\ . \tag{5.10}$$

Details on how to obtain such draws are in Appendix C.

Pure-model prediction If there are no calibration/tuning parameters, define the pure-model prediction of $y^R(\boldsymbol{x})$ simply as $\hat{y}^M(\boldsymbol{x})$. If we have available a new model run at input $\boldsymbol{x}$, then we do not need the approximation and can use $y^M(\boldsymbol{x})$; modelers often indeed plan to perform a new model run if a prediction is desired at a new $\boldsymbol{x}$. If there are calibration/tuning parameters, use an estimate $\hat{\boldsymbol{u}}$ based on the previous data, evaluate $\hat{y}^M$ (or y^M if possible) at input $(\boldsymbol{x}, \hat{\boldsymbol{u}})$ and define the pure-model prediction as $\hat{y}^M(\boldsymbol{x}, \hat{\boldsymbol{u}})$. For $\hat{\boldsymbol{u}}$, use the posterior mean or mode of $\boldsymbol{u}$, although other choices could be made. Denote the pure-model prediction by $\hat{\boldsymbol{y}}^M_{\text{NEW}}(\hat{\boldsymbol{u}})$.

For the pedagogic example, using the posterior mean $\hat{u} = 1.29$, the pure-model prediction at $t = 1$ is $\hat{y}^M(1, 1.29) = 1.43$. The entire pure-model prediction function is given in the upper left graph in Figure 3, for both $\hat{u}$ set equal to the posterior mean (dash-dotted line) and the least squares estimate (dotted line).

For spotweld, the entire pure-model prediction function $\hat{y}^M(L, C, G, \hat{u})$ – based on the computer model approximation and $\hat{u}$ the posterior mean – is (for four different values of load and gauge) graphed as the solid lines in the top graphs of Figure 5.

Bias-corrected prediction The bias-corrected prediction of the true process y^R at $\boldsymbol{x}$ is given by the estimate of the posterior predictive mean of $y^R(\boldsymbol{x})$, that is

$$\hat{\boldsymbol{y}}^R_{\text{NEW}} = \frac{1}{N} \sum_{i=1}^{N} \left[\hat{\boldsymbol{y}}^{M(i)}_{\text{NEW}} + \boldsymbol{b}^{(i)}_{\text{NEW}} \right]\ . \tag{5.11}$$

If the code is fast, the draw from the approximation to the code in the formula above is replaced by its actual value.

When bias is present, the bias-corrected prediction improves on the pure-model prediction. For example, in the pedagogic example, $\hat{y}^R(1) = 2.06$; the true value of the real process is $y^R(1) = 2.14$ and the pure-model prediction is 1.43. The entire bias-corrected prediction function is given as the solid line in the upper left graph in Figure 3. For spotweld, the entire bias-corrected prediction function, $\hat{y}^R(L, C, G)$, is (for four different values of load and gauge) graphed as the solid lines in the bottom graphs of Figure 5.

Bias of the pure-model prediction Since common practice today is to utilize some variant of pure-model prediction, it is useful to explicitly look at the bias of this procedure. The bias function of pure-model prediction is clearly given by

$$\hat{\boldsymbol{b}}_{\hat{u}} \equiv \hat{\boldsymbol{y}}^R_{\text{NEW}} - \boldsymbol{y}^M_{\text{NEW}}(\hat{\boldsymbol{u}})\,.$$

If one were actually trying to establish that the computer model is uniformly valid, in some sense, one would have to show that this bias function is effectively zero.

For the pedagogic example, the bias function for pure-model prediction is given in the solid line in the bottom two graphs of Figure 3 when, respectively, $\hat{u}$ is the posterior mean and least squares estimate. Clearly the bias is far from zero in either case. In spotweld, the bias function, $\hat{b}_{\hat{u}}(L, C, G)$, for pure-model prediction is graphed (for four different values of load and gauge) as the solid lines in the middle graphs of Figure 5.

Variances of these predictors The covariance matrices corresponding to the pure-model predictor and the bias-corrected predictor can be estimated, respectively, by

$$\begin{aligned}
\text{Cov}(\boldsymbol{y}^M_{\text{NEW}}(\hat{\boldsymbol{u}})) &= \frac{1}{N}\sum_{i=1}^{N}[\boldsymbol{y}^M_{\text{NEW}}(\hat{\boldsymbol{u}}) - (\hat{\boldsymbol{y}}^{M(i)}_{\text{NEW}} + \boldsymbol{b}^{(i)}_{\text{NEW}})][\boldsymbol{y}^M_{\text{NEW}}(\hat{\boldsymbol{u}}) - (\hat{\boldsymbol{y}}^{M(i)}_{\text{NEW}} + \boldsymbol{b}^{(i)}_{\text{NEW}})]' \\
\text{Cov}(\hat{\boldsymbol{y}}^R_{\text{NEW}}) &= \frac{1}{N}\sum_{i=1}^{N}[\hat{\boldsymbol{y}}^R_{\text{NEW}} - (\hat{\boldsymbol{y}}^{M(i)}_{\text{NEW}} + \boldsymbol{b}^{(i)}_{\text{NEW}})][\hat{\boldsymbol{y}}^R_{\text{NEW}} - (\hat{\boldsymbol{y}}^{M(i)}_{\text{NEW}} + \boldsymbol{b}^{(i)}_{\text{NEW}})]'\,.
\end{aligned}$$

It is easy to see that

$$\text{Cov}(\hat{\boldsymbol{y}}^R_{\text{NEW}}) = \text{Cov}(\boldsymbol{y}^M_{\text{NEW}}(\hat{\boldsymbol{u}})) - \hat{\boldsymbol{b}}_{\hat{u}}\ \hat{\boldsymbol{b}}'_{\hat{u}}$$

so that bias-corrected prediction will clearly have smaller variance than pure-model prediction (a strong incentive for use of bias-corrected prediction).

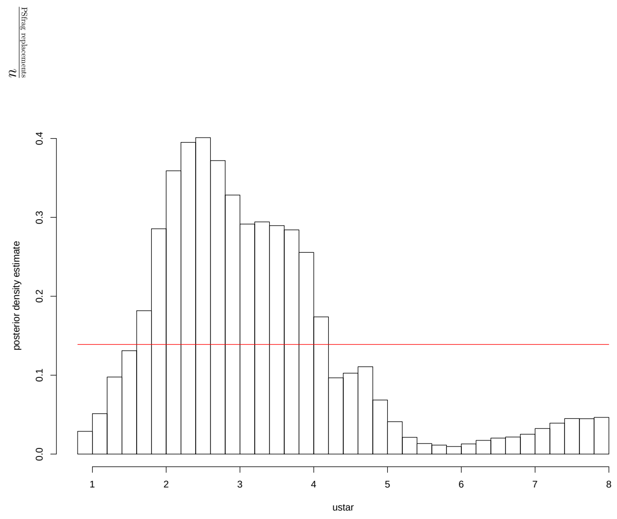


Figure 4: The posterior distribution of the tuning parameter u in the Spotweld example.

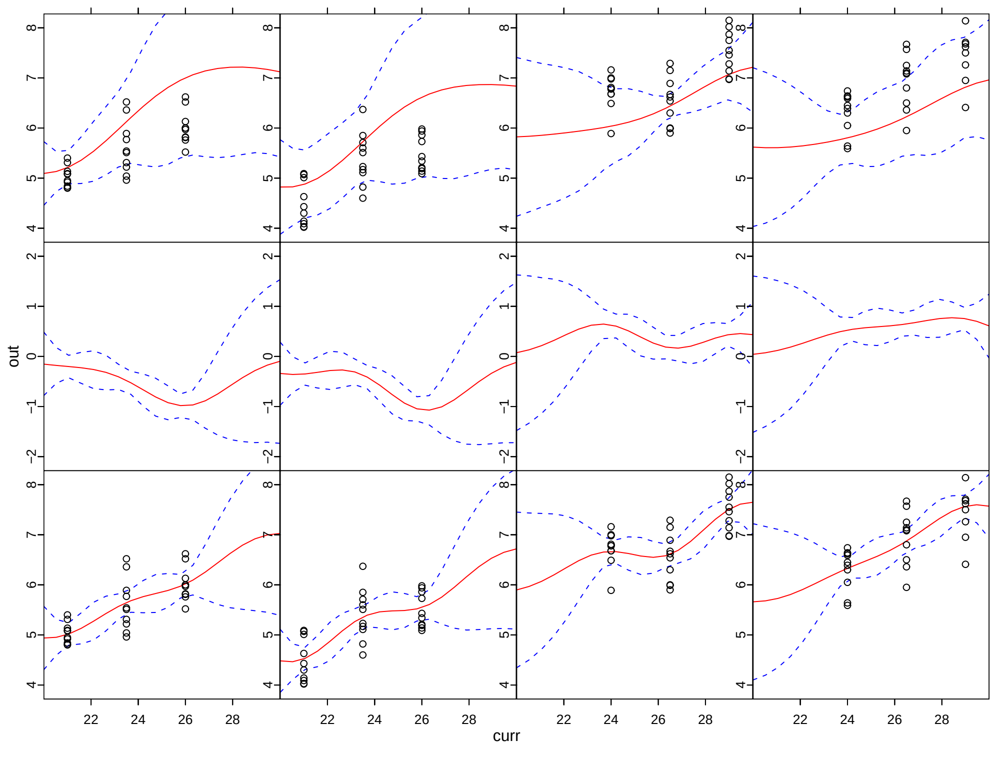


Figure 5: Spotweld Example: For four values of load and gauge, First Row: the pure-model weld diameter predictions, $\hat{y}^M(L, C, G, \hat{u})$, and 90% tolerance bands; Middle Row: the associated biases, $\hat{b}_{\hat{u}}(L, C, G)$, and 90% tolerance bands; Last Row: the bias-corrected predictions, $\hat{y}^R(L, C, G)$, and 90% tolerance bands. The circles represent the field data that were observed at those input values.

5.2.3 Tolerance bounds

As discussed in Section 1, we are primarily concerned with the predictive accuracy statement: "with probability γ, the prediction is within (tolerance) τ of the true $y^R(\boldsymbol{x})$." Such tolerance bounds for pure-model prediction are straightforwardly obtained from the samples (5.10). For a given γ we can estimate $\boldsymbol{\tau} = (\tau(\boldsymbol{x}) : \boldsymbol{x} \in D^F_{\text{NEW}})'$ by making sure that $\gamma \times 100\%$ of samples satisfy

$$\left| \hat{\boldsymbol{y}}^M_{\text{NEW}}(\hat{\boldsymbol{u}}) - \left[\hat{\boldsymbol{y}}^{M(i)}_{\text{NEW}} + \boldsymbol{b}^{(i)}_{\text{NEW}} \right] \right| < \boldsymbol{\tau} \,.$$

[In the previous formulae, all operations should be interpreted in a componentwise fashion, i.e.: $|\boldsymbol{x}| = (|x_i|, i = 1, \ldots, n)$ and $\boldsymbol{x} < \boldsymbol{y}$ iff $x_i < y_i$, $i = 1, \ldots, n$.]

Similarly, for the bias-corrected prediction the tolerance $\boldsymbol{\tau}$ is estimated by making sure that $\gamma \times 100\%$ of the samples satisfy

$$\left| \hat{\boldsymbol{y}}^R_{\text{NEW}} - \left[\hat{\boldsymbol{y}}^{M(i)}_{\text{NEW}} + \boldsymbol{b}^{(i)}_{\text{NEW}} \right] \right| < \boldsymbol{\tau} \,.$$

The tolerance bands for the bias of pure-model prediction follow from simply subtracting the pure-model prediction function, $\hat{y}^M(\boldsymbol{x}, \hat{\boldsymbol{u}})$, from the bands for bias-corrected prediction.

Pedagogic: The top-left panel of Figure 3 shows the bias-corrected predictions as a function of time (solid line) along with the (pointwise) 90% tolerance bounds (dashed lines). Also depicted are the pure-model predictions associated with the posterior mean (dash-dot line) and least-squares (dotted line) of u. These are plotted without tolerance bounds, which would be huge; there is little to be gained by depicting these – the message is clear that pure-model predictions are far from reality, while the bias-corrected predictions (and tolerance bounds) track reality very well.

The lower panel shows the estimate of the bias corresponding to each of the pure-model predictions, along with 90% credible intervals.

It can be convenient and straightforward – though we do not pursue the matter – to modify the definition of tolerance bounds by making them asymmetric and determine $(\boldsymbol{\tau}_1, \boldsymbol{\tau}_2)$ such that $\gamma \times 100\%$ of the predictive samples satisfy

$$\hat{\boldsymbol{y}}^{M(i)}_{\text{NEW}} + \boldsymbol{b}^{(i)}_{\text{NEW}} - \boldsymbol{\tau}_1 < \hat{\boldsymbol{y}}^R_{\text{NEW}} < \hat{\boldsymbol{y}}^{M(i)}_{\text{NEW}} + \boldsymbol{b}^{(i)}_{\text{NEW}} + \boldsymbol{\tau}_2 \,,$$

subject to minimizing $\boldsymbol{\tau}_1 + \boldsymbol{\tau}_2$ componentwise. This would be useful if bias is very large and the tolerance bounds would be one-sided or nearly so.

5.2.4 Comparison of Full Bayes and Modular-MLE analyses

The spotweld example was examined from a fully Bayesian perspective in Higdon et al. (2004). Although their prior specification is different from ours, the results are very close. Recall that the advantage of the approach here is that the sampling mechanism is quite stable and automatic and allows non-experts to directly apply the methodology with relatively simple code.

Also for comparison purposes, a full Bayesian (modular) analysis of the pedagogic example was implemented, using methodology in Bayarri et al. (2002) and Paulo (2005). The results are summarized in Figure 6. Comparing this figure with Figure 3 shows that the approaches yield qualitatively very similar answers. In particular, the bias-corrected predictions and tolerance bands are almost identical.

5.3 Confounding of tuning and bias

When $\boldsymbol{u}$ is present, there is confounding between $\boldsymbol{u}$ and the bias function; they are not identifiable. There appears to be some controversy about this – witness the discussion in Kennedy and O'Hagan (2001). To see the confounding, note again equation (5.7):

$$y^R(\boldsymbol{x}) = y^M(\boldsymbol{x}, \boldsymbol{u}) + b_{\boldsymbol{u}}(\boldsymbol{x}) \ .$$

Suppose that one observes a huge amount of field data so that (see (5.8)) $y^R(\boldsymbol{x})$ becomes effectively known. Suppose also that we can view $y^M(\boldsymbol{x}, \boldsymbol{u})$ as a completely known function. Then – even in this most favorable situation – for each $\boldsymbol{u}$, there is a $b_{\boldsymbol{u}}(\boldsymbol{x})$ that satisfies the above equation; they cannot separately be identified. Note that this happens because the support of the prior for $b_{\boldsymbol{u}}(\boldsymbol{x})$ is *any* (appropriately smooth) function; in contrast, if $b_{\boldsymbol{u}}(\boldsymbol{x})$ had a parametric form, identifiability could result, but it will rarely be appropriate in the context of analysis of complex computer models to assume that $b_{\boldsymbol{u}}(\boldsymbol{x})$ has a specified parametric form.

This general non-identifiability has rather severe implications for the possible methods of validation analysis. Bayesian analysis is clearly tenable, for several reasons:

- Predictions and attached uncertainties can be stable, even though separate inference on $\boldsymbol{u}$ and $b_{\boldsymbol{u}}(\boldsymbol{x})$ may be less stable, due to the confounding. As an illustration, consider the pedagogic example but with the GASP prior allowed to have a nonzero (unknown) mean μ^b. The results of this analysis are given in Figure 7, and should be compared with the analysis in Figure 3. The bias-corrected predictions change very little, even though the changes in the posterior for $\boldsymbol{u}$ and the bias were significant.

- If inference concerning a calibration parameter $\boldsymbol{u}$ is a primary goal, or if it is desired to use the model to predict outside the range of the data, it is crucial to incorporate good prior information about $\boldsymbol{u}$. Good prior information can prevent the tendency to 'overtune' $\boldsymbol{u}$, resulting in better inference and better out-of-sample prediction.

- Allowing greater flexibility in the bias function can also help control overtuning of $\boldsymbol{u}$. In Figure 7, for instance, the increased prior flexibility allowed for the bias resulted in the

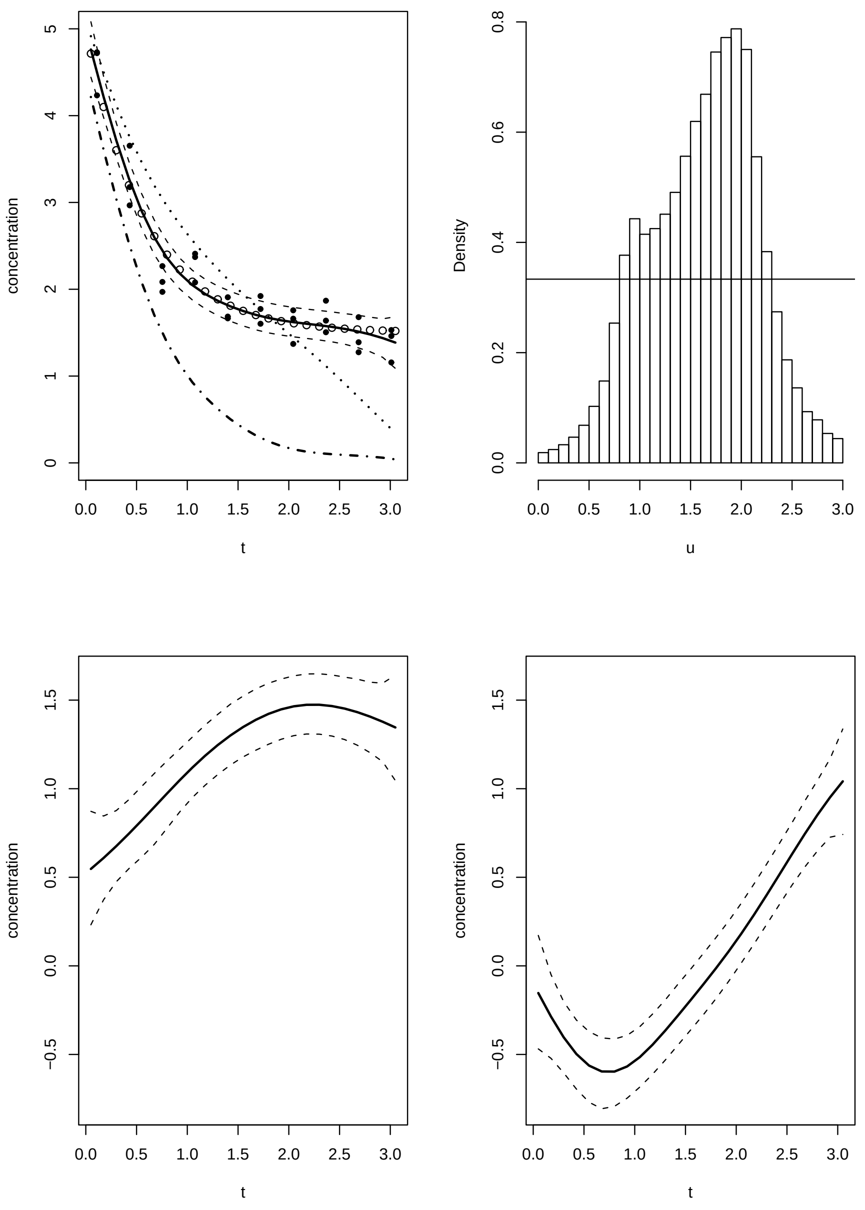


Figure 6: Pedagogic Example: The difference between this set of plots and those of Figure 3 is that these were produced using a full Bayesian analyis in each stage of the modular approach (Full-modular analysis). Top-Left Panel: The solid line is the bias-corrected prediction, the dashed lines are 90% tolerance bounds. The dash-dotted and dotted lines corresponds, respectively, to the pure-model prediction associated to the posterior mean and least-squares estimate of u. The circles are the true values of the real process, whereas the triangles represent the observed field data. Top-Right Panel: Posterior distribution of u. Lower Panels: Estimate of bias corresponding to the pure-model prediction associated to the posterior mean of u (left plot) and to the least-squares estimate of u (right plot).

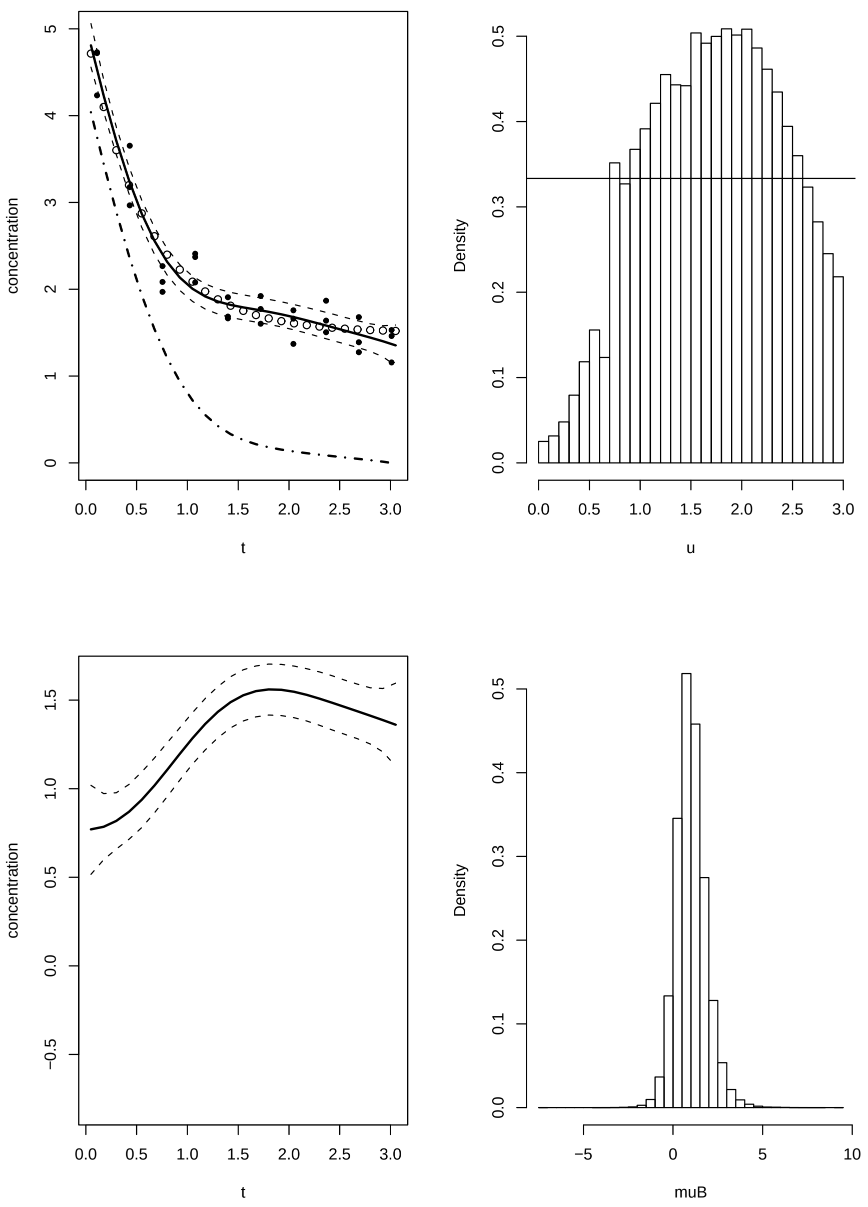


Figure 7: Pedagogic Example: The difference between this set of plots and those of Figure 3 is that these were produced allowing the bias function to have an unknown mean. Top-Left Panel: The solid line is the bias-corrected prediction, the dashed lines are 90% tolerance bounds. The dash-dotted line corresponds to the pure-model prediction associated to the posterior mean of u. The circles are the true values of the real process, whereas the triangles represent the observed field data. Top-Right Panel: Posterior distribution of u. Lower-Left Panel: Estimate of bias corresponding to the pure-model prediction associated to the posterior mean of u. Lower-Right Panel: Histogram of the posterior distribution of the bias mean, μ^b.

posterior having a larger bias, with less tuning of the calibration parameter $\boldsymbol{u}$. Indeed, the posterior mean of $\boldsymbol{u}$ moved to 1.72, which is essentially the correct value, although this may have just been a coincidence.

It is unclear how one would approach this crucial issue of confounding from a non-Bayesian perspective.

6 Feedback; Feed Forward (Step 6)

The analyses in Step 4 and Step 5 will contribute to the dynamic process of improving the model and updating the I/U map by identifying

- Model inputs whose uncertainties need to be reduced
- Needs (such as additional analyses and additional data) for closer examination of important regions or parts of the model
- Flaws that require changes in the model
- Revisions to the evaluation criteria.

In Spotweld, for instance, the posterior distribution of u (Figure 4) will now replace the uncertainty entry in the I/U map. Another aspect of feedback is use of the Step 4 and Step 5 analyses to further refine the validation process; e.g. to design additional validation experiments.

The feed-forward notion is to develop capability to predict the accuracy of new models that are related to models that have been studied, but for which no specific field data is available. This can be done through utilization of hierarchical Bayesian techniques, and we will explore it elsewhere.

7 Concluding Comments

We collect here some relevant comments that would have otherwise impeded the flow of the paper.

1. Combined Validation and Calibration

   One test bed example (pedagogic) had a calibration parameter, the other (spotweld) a tuning parameter. There is a distinction, although mathematically they are treated the same. Tuning uses field data to bring the model closer to reality while calibration is a process by which unknown model parameters are estimated from data. The distinction is that, in calibration, one tries to find the true – but unknown – physical value of a parameter, while in tuning one simply tries to find the best fitting value. The values of a tuning parameter will typically change if the data change and tuning may produce a good model for prediction in the range of the field data, but may well give very bad predictions outside this range.

It is generally believed that data used for calibration/tuning cannot simultaneously be used for model validation. However, the Bayesian methodology described readily accommodates such simultaneous use of data by incorporating the posterior distribution of the tuning parameters in the overall assessment of uncertainties. In contrast, simply replacing a tuning parameter by some optimal 'tuned' value $\hat{\boldsymbol{u}}$ (commonly done using least-squares) obscures the interaction between bias and tuning, and can lead to overly optimistic assessments of validity.

2. New Model Runs for Prediction

   In performing predictions, it is frequently sensible to include new model runs, if feasible, to obtain $y^M(\boldsymbol{x}, \hat{\boldsymbol{u}})$ for some key values of $\boldsymbol{x}$. In this paper we emphasized prediction when such new runs are unavailable, but the analysis can easily incorporate such new runs (cf. Appendix C), without having to redo all the computations from scratch. Utilization of such model runs may be particularly helpful in assessing changes arising from moving from input $\boldsymbol{x}$ to nearby input $\boldsymbol{x}'$. We forego further consideration of this modification.

References

Aslett, R., Buck, R. J., Duvall, S. G., Sacks, J., and Welch, W. J. (1998), "Circuit Optimization Via Sequential Computer Experiments: Design of an Output Buffer," *Applied Statistics*, 47, 31–48.

Bates, R. A., Buck, R. J., Riccomagno, E., and Wynn, H. P. (1996), "Experimental design and observation for large systems (Disc: p95-111)," *Journal of the Royal Statistical Society, Series B, Methodological*, 58, 77–94.

Bayarri, M., Berger, J., Garcia-Donato, G., Palomo, J., Sacks, J., Walsh, D., Cafeo, J., and Parthasarathy, R. (2005a), "Computer Model Validation with Function Output," Tech. rep., National Institute of Statistical Sciences.

Bayarri, M., Berger, J., Kennedy, M., Kottas, T., Paulo, R., Sacks, J., Cafeo, J., Lin, C., and Tu, J. (2005b), "Bayesian Validation of a Computer Model for Vehicle Collision," Tech. rep., National Institute of Statistical Sciences.

Bayarri, M. J., Berger, J. O., Higdon, D., Kennedy, M. C., Kottas, A., Paulo, R., Sacks, J., Cafeo, J. A., Cavendish, J., Lin, C. H., and Tu, J. (2002), "A Framework for Validation of Computer Models," Tech. Rep. 128, National Institute of Statistical Sciences, http://www.niss.org/technicalreports/tr128.pdf.

Berk, R., Bickel, P., Campbell, K., Fovell, R., Keller-McNulty, S., Kelly, E., Linn, R., Park, B., Perelson, A., Rouphail, N., Sacks, J., and Schoenberg, F. (2002), "Workshop on statistical approaches for the evaluation of complex computer models," *Statistical Science*, 17, 173–192.

Cafeo, J. and Cavendish, J. (2001), "A Framework For Verification And Validation Of Computer Models and Simulations," Internal general motors document, to be published, GM Research & Development Center.

Caswell, H. (1976), "The validation problem," in *Systems Analysis and Simulation in Ecology, vol. IV. B.C. Patten (ed.)*, Academic Press: New York, pp. 313–325.

Chapman, W., Welch, W., Bowman, K., Sacks, J., and Walsh, J. (1994), "Arctic sea ice variability: Model sensitivities and a multidecadal simulation," *J. of Geophysical Research*, 99, 919–935.

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

Craig, P. S., Goldstein, M., Seheult, A. H., and Smith, J. A. (1997), "Pressure matching for hydrocarbon reservoirs: a case study in the use of Bayes linear strategies for large computer experiments," in *Case Studies in Bayesian Statistics: Volume III. Gatsonis, C., Hodges, J. S., Kass, R. E., McCulloch, R., Rossi, P. and Singpurwalla, N. D. (eds.)*, pp. 36–;93.

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

Easterling, R. G. (2001), "Measuring the Predictive Capability of Computational Models: Principles and Methods, Issues and Illustrations," Tech. Rep. SAND2001-0243, Sandia National Laboratories.

Goldstein, M. and Rougier, J. C. (2003), "Calibrated Bayesian forecasting using large computer simulators," Tech. rep., Statistics and Probability Group, University of Durham, http://www.maths.dur.ac.uk/stats/physpred/papers/CalibratedBayesian.ps.

— (2004), "Probabilistic formulations for transferring inferences from mathematical models to physical systems," Tech. rep., Statistics and Probability Group, University of Durham, http://www.maths.dur.ac.uk/stats/physpred/papers/directSim.pdf.

Gough, W. and Welch, W. (1994), "Parameter space exploration of an ocean general circulation model using an isopycnal mixing parametrization," *Journal of Marine Research*, 52, 773–796.

Higdon, D., Kennedy, M. C., Cavendish, J., Cafeo, J., and Ryne, R. D. (2004), "Combining field data and computer simulations for calibration and prediction," *SIAM Journal on Scientific Computing*, 26, 448–466.

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

Kennedy, M. C., O'Hagan, A., and Higgins, N. (2002), "Bayesian analysis of computer code outputs," in *Quantitative Methods for Current Environmental Issues. C. W. Anderson, V. Barnett, P. C. Chatwin, and A. H. El-Shaarawi (eds.),* Springer-Verlag: London, pp. 227–243.

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

Oakley, J. (2004), "Estimating percentiles of computer code outputs," *Applied Statistics*, 53, 83–93.

Oakley, J. and O'Hagan, A. (2002), "Bayesian inference for the uncertainty distribution of computer model outputs," *Biometrika*, 89, 769–784.

— (2004), "Probabilistic sensitivity analysis of complex models: a Bayesian approach," *Journal of the Royal Statistical Society B*, 66, 751–769.

Oberkampf, W. and Trucano, T. (2000), "Validation Methodology in Computational Fluid Dynamics," Tech. Rep. 2000-2549, American Institute of Aeronautics and Astronautics.

Oreskes, N., Shrader-Frechette, K., and Belitz, K. (1994), "Verification, validation and confirmation of numerical models in the earth sciences," *Science*, 263(5147), 641–46.

Paulo, R. (2005), "Default priors for Gaussian processes," *Annals of Statistics*, in press.

Paulo, R., Lin, J., Rouphail, N., and Sacks, J. (2004), "Calibrating and Validating Deterministic Traffic Models: Application to the HCM Control Delay at Signalized Intersections," *Transportation Research Record: Journal of the Transportation Research Board*, to appear, earlier version available at http://www.niss.org/technicalreports/tr144.pdf.

Pilch, M., Trucano, T., Moya, J. L., Froehlich, G. Hodges, A., and Peercy, D. (2001), "Guidelines for Sandia ASCI Verification and Validation Plans - Content and Format: Version 2.0," Tech. Rep. SAND 2001-3101, Sandia National Laboratories.

Roache, P. (1998), *Verification and Validation in Computational Science and Engineering*, Albuquerque: Hermosa Publishers.

Robert, C. and Casella, G. (1999), *Monte Carlo Statistical Methods*, New York: Springer-Verlag.

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989), "Design and analysis of computer experiments (C/R: p423-435)," *Statistical Science*, 4, 409–423.

Saltelli, A., Chan, K., and Scott, M. (eds.) (2000), *Sensitivity Analysis*, Chichester: Wiley.

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

Stainforth, D. A., Aina, T., Christensen, C., Collins, M., Faull, N., Frame, D. J., Kettleborough, J. A., Knight, S., Martin, A., Murphy, J. M., Piani, C., Sexton, D., Smith, L. A., Spicer, R. A., Thorpe, A. J., and Allen, M. R. (2005), "Uncertainty in predictions of the climate response to rising levels of greenhouse gases," *Nature*, 433, 403–406.

Trucano, T., Pilch, M., and Oberkampf, W. O. (2002), "General Concepts for Experimental Validation of ASCII Code Applications," Tech. Rep. SAND 2002-0341, Sandia National Laboratories.

Wang, P. and Hayden, D. (1999), "Computational Modeling of Resistance Spot Welding of Aluminum," Tech. Rep. Research Report R&D - 9152, GM Research & Development Center.

Welch, W. J., Buck, R. J., Sacks, J., Wynn, H. P., Mitchell, T. J., and Morris, M. D. (1992), "Screening, predicting, and computer experiments," *Technometrics*, 34, 15–25.

A Summary of Notation and Assumptions

A.1 Gaussian process

A random process $y(\boldsymbol{z})$ is a Gaussian process with mean function $\mu(\cdot)$ and covariance function $C(\cdot,\cdot)$ if, for any finite set $\{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n\}$, $y(\boldsymbol{z}_1),\ldots,y(\boldsymbol{z}_n)$ is multivariate normal with mean vector $(\mu(\boldsymbol{z}_1),\ldots,\mu(\boldsymbol{z}_n))'$ and covariance matrix $[C(\boldsymbol{z}_i,\boldsymbol{z}_j)]_{ij}$.

A.2 Model inputs

We denote in general the input vector to a computer model by $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{u})$, where $\boldsymbol{x}$ is a p-vector of controllable and specified inputs (velocity, current, etc.), and $\boldsymbol{u}$ is a q-vector of calibration/tuning parameters.

A.3 Reality and bias

Given $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{u})$, $b(\boldsymbol{x})$, and $y^M(\boldsymbol{z})$,

$$y^R(\boldsymbol{x}) = y^M(\boldsymbol{z}) + b(\boldsymbol{x}) \ , \tag{A-1}$$

where $b(\cdot)$ is an unknown function with constant prior mean μ^b (often just chosen to be zero)

$$b(\boldsymbol{x}) \mid \mu^b, \boldsymbol{\theta}^b \sim \mathrm{GP}(\mu^b, \frac{1}{\lambda^b}\, c^b(\cdot,\cdot)) \; , \tag{A-2}$$

$$c^b(\boldsymbol{x}, \boldsymbol{x}^\star) = \prod_{i=1}^{p} \exp\{-\beta_i^b\, (x_i - x_i^\star)^2\} \; , \tag{A-3}$$

$$\boldsymbol{\theta}^b = (\lambda^b, \boldsymbol{\beta}^b) = (\lambda^b, \beta_1^b, \ldots, \beta_p^b) \; . \tag{A-4}$$

Note that we choose the exponents of $|x_i - x_i^\star|$ in the bias correlation function to be 2, reflecting the fact that we expect the bias to be a smooth function.

A.4 Field (error)

Given $y^R(\boldsymbol{x})$,

$$y^F(\boldsymbol{x}) = y^R(\boldsymbol{x}) + \epsilon^F(\boldsymbol{x}) \; , \tag{A-5}$$

where

$$\epsilon^F(\boldsymbol{x}) \mid \lambda^F \sim \mathrm{GP}(0, \frac{1}{\lambda^F}\, c^F(\cdot,\cdot)) \; , \tag{A-6}$$

$$c^F(\boldsymbol{x}, \boldsymbol{x}^\star) = 1_{\{\boldsymbol{x}=\boldsymbol{x}^\star\}} \; . \tag{A-7}$$

This is assuming that the field data arise via independent measurements and are unbiased (mean of $\epsilon^F = 0$).

A.5 Model data

The *design space* for the model data is $D^M = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m\}$, where $\boldsymbol{z}_i = (\boldsymbol{x}_i, \boldsymbol{u}_i)$, $i = 1, \ldots, m$. Let $\boldsymbol{y}^M = (y^M(\boldsymbol{z}_1), \ldots, y^M(\boldsymbol{z}_m))'$.

A.6 Field data

The *design space* for the field data is $D^F = \{\boldsymbol{x}_1^\star, \ldots, \boldsymbol{x}_n^\star\}$. The data consists of n_i replications taken at each point in D^F. Denoting these replications as $\{y_j^F(\boldsymbol{x}_i^\star),\ j = 1, \ldots, n_i\}$, given $y^R(\boldsymbol{x}_i^\star)$, $j = 1, \ldots, n_i$, we can replace the field data with the independent sufficient statistics $\bar{\boldsymbol{y}}^F = (\bar{y}^F(\boldsymbol{x}_1^\star), \ldots, \bar{y}^F(\boldsymbol{x}_n^\star))'$, where $\bar{y}^F(\boldsymbol{x}_i^\star) = \frac{1}{n_i}\sum_{j=1}^{n_i} y_j^F(\boldsymbol{x}_i^\star)$, and $s_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{n_i}[y_j^F(\boldsymbol{x}_i^\star) - \bar{y}^F(\boldsymbol{x}_i^\star)]^2$.

These are such that, independently,

$$\bar{y}^F(\boldsymbol{x}_i^\star) = y^R(\boldsymbol{x}_i^\star) + \bar{\epsilon}^F(\boldsymbol{x}_i^\star) \;, \tag{A-8}$$

$$\bar{\epsilon}^F(\boldsymbol{x}_i^\star) \mid \lambda^F, \boldsymbol{\theta}^T \sim \mathrm{GP}(0, \frac{1}{\lambda^F}\, \bar{c}^F(\cdot,\cdot)) \;, \tag{A-9}$$

$$\bar{c}^F(\boldsymbol{x}_i^\star, \boldsymbol{x}_j^\star) = (n_i n_j)^{-1/2} c^F(\boldsymbol{x}_i^\star, \boldsymbol{x}_j^\star) \;, \tag{A-10}$$

$$s_F^2 \mid \lambda^F \sim \tfrac{1}{\lambda^F}\; \chi^2(\textstyle\sum_{i=1}^n (n_i - 1)) \;. \tag{A-11}$$

A.7 Likelihood

We present the more complicated case of a slow computer model, when the approximation detailed in Section 4 must be used. The situation where y^M is fast follows as a particular case.

Define $C^f(D^g, D^h)$ to be the matrix with (i,j) entry $c^f(\boldsymbol{w}_i, \boldsymbol{w}_j^\star)$, and $\mu^f(D^f)$ to be the vector with component i equal to $\mu^f(\boldsymbol{w}_i)$, where $\boldsymbol{w}_i$ and $\boldsymbol{w}_j^\star$ are, respectively, the i^{th} and j^{th} points in the design spaces D^g and D^h. Also, let $C^f(D^g, D^g) \equiv C^f(D^g)$.

Because the model run input values and field run input values need not coincide, definition of the likelihood requires formal augmentation of the field design space by the calibration parameters $\boldsymbol{u}$. When this is needed, we denote the augmented design space by $D_{\boldsymbol{u}}^F$ (which is the same design space as D^F, except that we simply replace $\boldsymbol{x}_i^\star$ by $(\boldsymbol{x}_i^\star, \boldsymbol{u})$). It is useful to augment the observed data ($\boldsymbol{y}^M$, $\bar{\boldsymbol{y}}^F$, s_F^2) with the bias function evaluated at D^F, $\boldsymbol{b}$, and the model values evaluated at points in $D_{\boldsymbol{u}}^F$, to be denoted $\boldsymbol{y}_\star^M$. Then, with the above in mind, we have

$$f(\bar{\boldsymbol{y}}^F, s_F^2, \boldsymbol{b}, \boldsymbol{y}_\star^M, \boldsymbol{y}^M \mid \boldsymbol{\theta}^L, \boldsymbol{\theta}^M, \mu^b, \boldsymbol{\theta}^b, \lambda^F, \boldsymbol{u}) = f(s_F^2 \mid \lambda^F) \;\times\; f(\bar{\boldsymbol{y}}^F \mid \boldsymbol{b}, \boldsymbol{y}_\star^M, \lambda^F) \;\times$$
$$f(\boldsymbol{b} \mid \boldsymbol{\theta}^b, \mu^b) \;\times\; f(\boldsymbol{y}_\star^M \mid \boldsymbol{y}^M, \boldsymbol{\theta}^L, \boldsymbol{\theta}^M, \boldsymbol{u}) \;\times\; f(\boldsymbol{y}^M \mid \boldsymbol{\theta}^L, \boldsymbol{\theta}^M) \;, \tag{A-12}$$

where

$$f(s_F^2 \mid \lambda^F) = \lambda^F\; \chi^2(\lambda^F\, s_F^2 \mid \textstyle\sum_{i=1}^n (n_i - 1)) \tag{A-13}$$

$$f(\bar{\boldsymbol{y}}^F \mid \boldsymbol{b}, \boldsymbol{y}_\star^M, \lambda^F) = \mathrm{N}(\bar{\boldsymbol{y}}^F \mid \boldsymbol{y}_\star^M + \boldsymbol{b}, \tfrac{1}{\lambda^F}\, \bar{C}^F(D^F, D^F)) \tag{A-14}$$

$$f(\boldsymbol{b} \mid \boldsymbol{\theta}^b, \mu^b) = \mathrm{N}(\boldsymbol{b} \mid \mu^b(D^F), \tfrac{1}{\lambda^b}\, C^b(D^F, D^F)) \tag{A-15}$$

$$f(\boldsymbol{y}_\star^M \mid \boldsymbol{y}^M, \boldsymbol{\theta}^L, \boldsymbol{\theta}^M, \boldsymbol{u}) = \mathrm{N}(\boldsymbol{y}_\star^M \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{A-16}$$

$$f(\boldsymbol{y}^M \mid \boldsymbol{\theta}^L, \boldsymbol{\theta}^M) = \mathrm{N}(\boldsymbol{y}^M \mid \mu^M(D^M),\; C^M(D^M, D^M)) \;, \tag{A-17}$$

with

$$\boldsymbol{\mu} = \mu^M(D_{\boldsymbol{u}}^F) + C^M(D_{\boldsymbol{u}}^F, D^M)\; [C^M(D^M, D^M)]^{-1}(\boldsymbol{y}^M - \mu^M(D^M))$$
$$\boldsymbol{\Sigma} = C^M(D_{\boldsymbol{u}}^F) - C^M(D_{\boldsymbol{u}}^F, D^M)\; [C^M(D^M, D^M)]^{-1} C^M(D^M, D_{\boldsymbol{u}}^F) \;.$$

Note that one can analytically integrate out $\boldsymbol{b}$ and $\boldsymbol{y}_\star^M$ in (A-12) to obtain the following expression for the marginal likelihood for the indicated parameters:

$$
\begin{aligned}
f(\bar{\boldsymbol{y}}^F, s_F^2, \boldsymbol{y}^M \mid \boldsymbol{\theta}^L, \boldsymbol{\theta}^M, \mu^b, \boldsymbol{\theta}^b, \lambda^F, \boldsymbol{u}) &= f(s_F^2 \mid \lambda^F)\times \\
&\mathrm{N}(\bar{\boldsymbol{y}}^F \mid \boldsymbol{\mu} + \mu^b(D^F), \boldsymbol{\Sigma} + \tfrac{1}{\lambda^F}\, \bar{C}^F(D^F, D^F) + \tfrac{1}{\lambda^b}\, C^b(D^F, D^F)) \ \times\ f(\boldsymbol{y}^M \mid \boldsymbol{\theta}^L, \boldsymbol{\theta}^M)\ . \quad \text{(A-18)}
\end{aligned}
$$

To complete the Bayesian analysis, priors must be chosen for the unknown parameters: those we recommend are given in Appendix A.9. The MCMC sampling scheme we recommend to sample from the posterior is given in Appendix B.

A.8 Modularization

Here we describe, in detail, the approximate Bayesian analysis which we refer to as the Modular Approach. The basic idea is to first do the analysis of all the model data, ignoring the contribution of the field data in estimating GASP model approximation parameters (including $\boldsymbol{\theta}^L$.) Then, treat the model parameters (other than tuning parameters) as specified by the resulting posterior distribution – or, possibly, by their maximum likelihood estimates – and incorporate the field data by a separate Bayesian analysis. Formally, this is a partial likelihood approach, treating (A-17) as the only part of the likelihood that is used to determine the model GASP parameters.

This approach is implemented as follows.

Stage 1 : Analyze the model data in isolation, to obtain the posterior density $p(\boldsymbol{\theta}^L, \boldsymbol{\theta}^M \mid \boldsymbol{y}^M)$, using (A-17) together with the prior density $p(\boldsymbol{\theta}^L, \boldsymbol{\theta}^M)$ specified in Appendix A.9. This will typically be represented by an MCMC cloud of realizations of points $(\boldsymbol{\theta}^L, \boldsymbol{\theta}^M)$.

Alternatively, if the MLE plug-in approach is used, simply utilize $(\hat{\boldsymbol{\theta}}^L, \hat{\boldsymbol{\theta}}^M)$ in the following.

Stage 2 : To incorporate the field data $\boldsymbol{y}^F$, find the marginal posterior (defining $\boldsymbol{\theta} = (\boldsymbol{\theta}^L, \boldsymbol{\theta}^M)$)

$$
p(\mu^b, \boldsymbol{\theta}^b, \lambda^F, \boldsymbol{u} \mid \boldsymbol{y}^F, \boldsymbol{y}^M, \text{Stage1}) = \int p(\mu^b, \boldsymbol{\theta}^b, \lambda^F, \boldsymbol{u} \mid \boldsymbol{y}^F, \boldsymbol{y}^M, \boldsymbol{\theta})\ p(\boldsymbol{\theta} \mid \boldsymbol{y}^M)\, d\boldsymbol{\theta}
$$

(or utilize $p(\mu^b, \boldsymbol{\theta}^b, \lambda^F, \boldsymbol{u} \mid \boldsymbol{y}^F, \boldsymbol{y}^M, \hat{\boldsymbol{\theta}})$ if the MLE plug-in approach is used). Note that, depending on the application, one might use the posterior which includes $\boldsymbol{b}$ and $\boldsymbol{y}^M_\star$, if that simplifies the sampling mechanism.

This step is implemented by drawing a point from the Stage 1 cloud (or utilizing $(\hat{\boldsymbol{\theta}}^L, \hat{\boldsymbol{\theta}}^M)$), generating $\mu^b, \boldsymbol{\theta}^b, \lambda^F, \boldsymbol{u}$ (and perhaps also $\boldsymbol{b}$ and $\boldsymbol{y}^M_\star$), and repeating. Note that, in generating from $p(\mu^b, \boldsymbol{\theta}^b, \lambda^F \mid \boldsymbol{y}^F, \boldsymbol{y}^M, \boldsymbol{\theta}, \boldsymbol{u})$, the full likelihood ((A-18) or (A-12)) must be used, together with the prior density $p(\mu^b, \boldsymbol{\theta}^b, \lambda^F)\ p(\boldsymbol{u})$.

The motivation and advantages of the modular approach are as follows.

1. Field data can affect the GASP model approximation parameters (the α's, β's, and λ's) in undesirable ways, allowing them to do some of the 'tuning' of the model, instead of limiting the tuning effect of the field data to $\boldsymbol{u}$. Indeed, this was observed in the spotweld example, where $\boldsymbol{u}$ was shifted to the edge of its domain, and the GASP parameters played the role of model 'tuners.' The modular approach prevents this from happening.

2. This easily generalizes to systems with several model components, M_i, each of which has separate model-run data. Dealing first with the separate model-run data, in setting up the GASP model approximations, and incorporating the field data only at the tuning/validation stage, makes for an easier-to-understand and computationally much more efficient process.

3. Computations are considerably simplified, since the overall posterior factors into lower dimensional blocks.

A.9 Prior distributions

Paulo (2005) specifically addresses the problem of specifying the prior $p(\boldsymbol{\theta}^L, \boldsymbol{\theta}^M)$ and of sampling from the corresponding posterior $p(\boldsymbol{\theta}^L, \boldsymbol{\theta}^M \mid \boldsymbol{y}^M)$. In that paper, several priors are derived and compared on the basis of their frequentist properties.

However, as already mentioned, for computational reasons we recommend simply computing the maximum likelihood estimates of $\boldsymbol{\theta}^L, \boldsymbol{\theta}^M$ based on model data alone, and consider those parameters as fixed in the second stage of the modular approach.

In order to carry out the analysis of the second stage, one must specify the prior on μ^b, $\boldsymbol{\theta}^b = (\boldsymbol{\beta}^b, \lambda^b)$, λ^F, and $\boldsymbol{u}$. The prior on the calibration parameter $\boldsymbol{u}$ is the one specified in the I/U map. Choosing default priors for the other bias GASP parameters is actually quite challenging, because of the typically limited data that is available, and the fact that there is no direct data about the bias. Also, as with the model GASP parameters, we noticed considerable confounding between the parameters, and thus opted for a method (described below) that fixes the $\boldsymbol{\beta}^b$ parameters at reasonable values, and allows only λ^b (and possibly μ^b) to vary.

It then remains to choose priors for λ^b and λ^F. As long as replications are available, use of a standard prior such as $1/\lambda^F$ should be fine for the error precision, but replications are not always available. Other problems are that the likelihood for λ^b can be quite flat, and λ^b can be highly confounded with $\mathbf{u}$. This leads us below to advocate use of data-dependent priors, centered at estimates of λ^b and λ^F.

Any of these choices can be criticized from a strict Bayesian viewpoint, but we feel that there are compelling practical reasons to make them. First, there is a great deal of confounding going on here; we want the flexibility of GASPs in approximating the model and representing the bias, but they have too many parameters. Proper subjective priors for these parameters are simply not going to be available, and the principled objective priors of Paulo (2005) are computationally too intensive. Since the methodology is being designed for use by non-experts, it also is not feasible to utilize more standard default priors with the advice to "watch out for convergence or stability issues." Finally, even with the rather *ad hoc* methods we use to determine the GASP parameters (and center some of the priors), the variability of the resulting predictions seems to be similar to that from a full (careful) Bayesian analysis. Hence we feel that we are capturing the major uncertainties of the problem, while using a blend of techniques that results in a reliable and stable methodology.

Here are the details of the proposed implementation:

1. Using the first stage approximation to the computer model, produce the pure-model prediction at the points in the field design space D^F augmented with a reasonable guess, $\tilde{\boldsymbol{u}}$, of the calibration parameter (e.g., the MLE or simply the apriori mean); recall that we denote this augmented design space by $D^F_{\tilde{\boldsymbol{u}}}$. Denote the vector of resulting model predictions by $\tilde{\boldsymbol{y}}^M$.

2. Treat the vector $\boldsymbol{y}^F - \tilde{\boldsymbol{y}}^M$ as a realization from a Gaussian process with a nugget, namely, as a realization from a multivariate Normal with constant mean μ^b and covariance matrix

$$\tfrac{1}{\lambda^b}\, C^b(D^F, D^F) + \tfrac{1}{\lambda^F}\, \mathbf{I}\ .$$

   Using the Welch GASP software, one can then obtain an (MLE) estimate of $\boldsymbol{\beta}^b$, which will be the fixed value used in the analysis. Note that, if the model and field design points were the same, there would be no need to use the model approximation to determine the vector $\tilde{\boldsymbol{y}}^M$.

3. The GASP software will also yield MLE estimates, $\hat{\lambda}^b$ and $\hat{\lambda}^F$, but it is important to allow λ^b and λ^F to vary in the Bayesian analysis. For these parameters, we choose independent Exponential priors with means equal to a modest multiple (e.g., 5) of the MLE's. In line with Paulo (2005), experience has shown that the final predictions are relatively insensitive to the choice of the multiplying factor.

4. If a nonzero mean, μ^b, is used for the bias, we suggest simply using the usual constant prior (which can be shown to yield a proper posterior). We typically do not use a mean for the bias, however, i.e. we usually set $\mu^b = 0$.

B The MCMC Method for Posterior Inference

Here we present the details of the MCMC method for posterior inference under the MLE-Modular approach, that which we recommend for routine implementation of the methodology. (When performing a full Bayesian analysis, algorithms described in Paulo (2005) and Bayarri et al. (2002) work well, although they may require monitoring and tuning.)

As detailed in Appendix A.9, the only parameters that have not been fixed are the calibration parameter $\boldsymbol{u}$, the precisions λ^F and λ^b, and possibly the bias mean μ^b. These are sampled in the

MCMC as follows: given the current state of the chain $\boldsymbol{y}^M_{\star\text{old}}, \boldsymbol{b}_{\text{old}}, \lambda^F_{\text{old}}, \lambda^b_{\text{old}}, \boldsymbol{u}_{\text{old}}$, we compute the next state as follows:

1. Generate $(\boldsymbol{y}^M_{\star\text{new}}, \boldsymbol{b}_{\text{new}})$ directly from its full conditional,

$$[\boldsymbol{y}^M_\star, \boldsymbol{b} \mid \lambda^b_{\text{old}}, \lambda^F_{\text{old}}, \boldsymbol{u}_{\text{old}}, \boldsymbol{y}^M, \bar{\boldsymbol{y}}^F, s^2_F] \tag{B-1}$$

   which is a Multivariate Normal whose parameters are determined using standard Kalman filter formulas – see the discussion below.

2. Generate λ^F_{new} from its full conditional

$$[\lambda^F \mid \lambda^b_{\text{old}}, \boldsymbol{u}_{\text{old}}, \boldsymbol{y}^M_{\star\text{new}}, \boldsymbol{b}_{\text{new}}, \boldsymbol{y}^M, \bar{\boldsymbol{y}}^F, s^2_F] = \Gamma(\lambda^F \mid a_1, a_2)$$

   where

$$\begin{aligned} a_1 &= \textstyle\sum_{i=1}^n n_i/2 + \alpha_F \\ a_2 &= r_F + s^2_F/2 + (\bar{\boldsymbol{y}}^F - \boldsymbol{b}_{\text{new}} - \boldsymbol{y}^M_{\star\text{new}})' \ \textbf{diag}\ \boldsymbol{n}\ (\bar{\boldsymbol{y}}^F - \boldsymbol{b}_{\text{new}} - \boldsymbol{y}^M_{\star\text{new}})/2 \end{aligned}$$

   and, *a priori*, $\lambda^F \sim \Gamma(\alpha_F, r_F)$ (in Appendix A.9 we recommended $\alpha_F = 1$ and $r_F = 5\hat{\lambda}^F$, but the MCMC works in this more general setting.)

3. Generate λ^b_{new} directly from its full conditional

$$[\lambda^b \mid \lambda^F_{\text{new}}, \boldsymbol{u}_{\text{old}}, \boldsymbol{y}^M_{\star\text{new}}, \boldsymbol{b}_{\text{new}}, \boldsymbol{y}^M, \bar{\boldsymbol{y}}^F, s^2_F] = \Gamma(\lambda^b \mid a_1, a_2)$$

   where

$$\begin{aligned} a_1 &= n/2 + \alpha_b \\ a_2 &= r_F + \boldsymbol{b}'_{\text{new}}\ [C^b(D^F, D^F)]^{-1}\ \boldsymbol{b}_{\text{new}}/2 \end{aligned}$$

   and, *a priori*, $\lambda^b \sim \Gamma(\alpha_b, r_b)$ (in Appendix A.9 we recommended $\alpha_b = 1$ and $r_b = 5\hat{\lambda}^b$, but the MCMC works in this more general setting.)

   Note that, if a nonzero bias mean is utilized in the analysis, one also has to sample μ^b_{new} directly from its full conditional

$$[\mu^b \mid \lambda^b_{\text{new}}, \lambda^F_{\text{new}}, \boldsymbol{u}_{\text{old}}, \boldsymbol{y}^M_{\star\text{new}}, \boldsymbol{b}_{\text{new}}, \boldsymbol{y}^M, \bar{\boldsymbol{y}}^F, s^2_F] = \text{N}(m, s^2)$$

where

$$m = \mathbf{1}' \, [C^b(D^F, D^F)]^{-1} \boldsymbol{b}_{\text{new}} / \mathbf{1}' \, [C^b(D^F, D^F)]^{-1} \mathbf{1}$$
$$s^{-2} = \lambda^b_{\text{new}} \mathbf{1}' \, [C^b(D^F, D^F)]^{-1} \mathbf{1} \, ,$$

and all the other steps should be carried out using the most recent value of μ^b.

4. Generate $\boldsymbol{u}_{\text{new}}$ using a Metropolis-Hastings step. We have had success with the strategy of choosing with probability Q (e.g., 0.5) to propose a draw from the prior and with probability $1 - Q$ to propose a locally perturbed version of the previous current value of the chain. The algorithm is thus

   Draw $q \sim \text{U}(0, 1)$ and

   (a) if $q < Q$, draw $\boldsymbol{v}$ from the prior on $\boldsymbol{u}$, $p(\boldsymbol{u})$,

   (b) else, draw $\boldsymbol{v} \sim \prod_i \text{U}(u_{i,\text{old}} - \epsilon_i, u_{i,\text{old}} + \epsilon_i)$, where $u_{i,\text{old}}$ is the ith component of $\boldsymbol{u}_{\text{old}}$ and the ϵ_i are chosen, say, as a fixed proportion of the range of u_i in the prior.

   Finally, set
$$\boldsymbol{u}_{\text{new}} = \begin{cases} \boldsymbol{v} & \text{w.p. } \rho(\boldsymbol{u}_{\text{old}}, \boldsymbol{v}) \\ \boldsymbol{u}_{\text{old}} & \text{w.p. } 1 - \rho(\boldsymbol{u}_{\text{old}}, \boldsymbol{v}) \end{cases}$$

   where

$$\rho(\boldsymbol{u}, \boldsymbol{v}) = \min \left\{ 1, \frac{f(\boldsymbol{y}^M_\star \mid \boldsymbol{y}^M, \boldsymbol{v})}{f(\boldsymbol{y}^M_\star \mid \boldsymbol{y}^M, \boldsymbol{u})} \frac{Q + (1 - Q) \, [\prod_i \text{U}(v_i \mid u_i - \epsilon_i, u_i + \epsilon_i)] / p(\boldsymbol{u})}{Q + (1 - Q) \, [\prod_i \text{U}(u_i \mid v_i - \epsilon_i, v_i + \epsilon_i)] / p(\boldsymbol{v})} \right\} \, . \tag{B-2}$$

   In (B-2), $f(\boldsymbol{y}^M_\star \mid \boldsymbol{y}^M, \boldsymbol{v})$ is as in (A-16) but with $(\boldsymbol{\theta}^L, \boldsymbol{\theta}^M)$ replaced by the estimates computed as described in Appendix A.9.

Regarding the parameters of the full conditional of the latent data, it is clear that $(\boldsymbol{y}^M_\star, \boldsymbol{b}, \boldsymbol{y}^M, \bar{\boldsymbol{y}}^F)$, conditional on all other parameters, is Multivariate Normal with mean

$$(\mu^M(D^F_{\boldsymbol{u}}), \mu^b(D^F), \mu^M(D^M), \mu^b(D^F) + \mu^M(D^F_{\boldsymbol{u}})) \tag{B-3}$$

and covariance matrix

$$\begin{pmatrix} \frac{1}{\lambda^M} C^M(D^F_{\boldsymbol{u}}) & & & \\ \mathbf{0} & \frac{1}{\lambda^b} C^b(D^F) & & \\ \frac{1}{\lambda^M} C^M(D^M, D^F_{\boldsymbol{u}}) & \mathbf{0} & \frac{1}{\lambda^M} C^M(D^M) & \\ \frac{1}{\lambda^M} C^M(D^F_{\boldsymbol{u}}) & \frac{1}{\lambda^b} C^b(D^F) & \frac{1}{\lambda^M} C^M(D^F_{\boldsymbol{u}}, D^M) & \frac{1}{\lambda^M} C^M(D^F_{\boldsymbol{u}}) + \frac{1}{\lambda^b} C^b(D^F) + \frac{1}{\lambda^F} \bar{C}^F(D^F) \end{pmatrix} . \tag{B-4}$$

As such, using standard Kalman filter formulas one can easily compute the parameters of the ensuing conditional distribution of $(\boldsymbol{y}^M_\star, \boldsymbol{b})$. One can draw from this joint distribution in several different ways, but direct simulation typically seems to work well.

C Predictions

For prediction, it is necessary to sample from the posterior predictive distribution of the real process evaluated at a set D^F_{NEW} of new design points. Assuming that the available field data is $\bar{\boldsymbol{y}}^F, s^2_F$ and that the available computer model data is $\boldsymbol{y}^M$, what we want is to obtain draws from

$$\int p(\{\hat{y}^M(\boldsymbol{x},\boldsymbol{u}), b(x) : \boldsymbol{x} \in D^F_{\text{NEW}}\} \mid \bar{\boldsymbol{y}}^F, s^2_F, \boldsymbol{y}^M, \boldsymbol{\theta})\; p(\boldsymbol{\theta} \mid \bar{\boldsymbol{y}}^F, s^2_F, \boldsymbol{y}^M)\; d\boldsymbol{\theta}$$

where $\boldsymbol{\theta} \equiv \{\boldsymbol{u}, \boldsymbol{\theta}^L, \boldsymbol{\theta}^M, \mu^b, \lambda^b, \boldsymbol{\beta}^b, \theta^b\}$. We have denoted those draws by (5.10). To obtain these, we proceed as follows: for each element of a sample from the posterior distribution of $\boldsymbol{\theta}$, $p(\boldsymbol{\theta} \mid \bar{\boldsymbol{y}}^F, s^2_F, \boldsymbol{y}^M)$, say $\boldsymbol{\theta}^{(i)}$, which can be obtained as detailed in Appendix B, we have to generate a realization from $p(\{\hat{y}^M(\boldsymbol{x},\boldsymbol{u}), b(x) : \boldsymbol{x} \in D^F_{\text{NEW}}\} \mid \bar{\boldsymbol{y}}^F, s^2_F, \boldsymbol{y}^M, \boldsymbol{\theta}^{(i)})$. This distribution is similar to (B-1) with the obvious modifications; hence, it is multivarite normal with mean vector and covariance matrix obtained using the standard Kalman filter formulas. The parameters for the joint normal are similar to (B-3) and (B-4); the mean is

$$(\mu^M(D^F_{\text{NEW}\ \boldsymbol{u}^{(i)}}), \mu^b(D^F_{\text{NEW}}), \mu^M(D^M), \mu^b(D^F) + \mu^M(D^F_{\boldsymbol{u}})) \tag{C-5}$$

and the covariance matrix follows similarly.

If one decides to collect more computer model data to aid prediction, formally one should rerun the MCMC to update the posterior of the unknown parameters given this additional information. That is rarely practical, even if one is following a modular approach, so we recommend to add the additional code data to the vector $\boldsymbol{y}^M$ but to leave all other aspects of the posterior unchanged.