Available online at www.sciencedirect.com
SCIENCE DIRECT®
Reliability Engineering and System Safety 91 (2006) 1331–1357
RELIABILITY ENGINEERING & SYSTEM SAFETY
www.elsevier.com/locate/ress

Calibration, validation, and sensitivity analysis: What's what

T.G. Trucano[a,*], L.P. Swiler[a], T. Igusa[b], W.L. Oberkampf[c], M. Pilch[c]

[a] *Optimization and Uncertainty Estimation Department, Sandia National Laboratories, P.O. Box 5800, Albuquerque, NM 87185-0819, USA*
[b] *Johns Hopkins University, Baltimore, MD 21218, USA*
[c] *Validation and Uncertainty Estimation Processes, Sandia National Laboratories, P.O. Box 5800, Albuquerque, NM 87185-0819, USA*

Available online 19 January 2006

Abstract

One very simple interpretation of calibration is to adjust a set of parameters associated with a computational science and engineering code so that the model agreement is maximized with respect to a set of experimental data. One very simple interpretation of validation is to quantify our belief in the predictive capability of a computational code through comparison with a set of experimental data. Uncertainty in both the data and the code are important and must be mathematically understood to correctly perform both calibration and validation. Sensitivity analysis, being an important methodology in uncertainty analysis, is thus important to both calibration and validation. In this paper, we intend to clarify the language just used and express some opinions on the associated issues. We will endeavor to identify some technical challenges that must be resolved for successful validation of a predictive modeling capability. One of these challenges is a formal description of a "model discrepancy" term. Another challenge revolves around the general adaptation of abstract learning theory as a formalism that potentially encompasses both calibration and validation in the face of model uncertainty.

1. Introduction

Our primary goal for this paper is to explore and differentiate the principles of calibration and validation for computational science and engineering (CS&E), as well as to present some related technical issues that are important and of current interest to us. Our conclusion is that calibration and validation are essentially different. To explain what we mean by calibration and validation, we restrict our attention to CS&E software systems, called *codes* here. We then define the product (output) of the execution of a code for a given choice of input to be the resulting *calculation*. Now, one definition of *calibration* is to adjust a set of code input parameters associated with one or more calculations so that the resulting agreement of the code calculations with a chosen and fixed set of experimental data is maximized (this requires a quantitative specification of the agreement). Compare this with the following simple definition of *validation*: that is, to quantify our confidence in the predictive capability of a code for a given application through comparison of calculations with a set of experimental data.

The foundation of our discussion below elaborates the meaning of these definitions of validation and calibration, primarily through the introduction of some mathematical formalism. Our formalism allows us to reasonably precisely argue that CS&E validation and calibration require rigorous comparison with *benchmarks*, which we precisely define in Section 2. Our discussion leads us to consider other concepts as well, including *uncertainty*, *prediction*, and *verification*, and their relationship to validation and calibration. Verification is a particularly important concept in CS&E and inevitably influences calibration and validation. We will explain why this is the case, and claim as well that validation and calibration in CS&E both *depend* on results of verification. We also claim that calibration is logically dependent on the results of validation, which is one way of emphasizing that calibration cannot be viewed as an adequate substitute for validation in many CS&E applications.

*Corresponding author.
E-mail addresses: tgtruca@sandia.gov (T.G. Trucano), lpswile@sandia.gov (L.P. Swiler), tigusa@jhu.edu (T. Igusa), wloberk@sandia.gov (W.L. Oberkampf), mpilch@sandia.gov (M. Pilch).

0951-8320/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ress.2005.11.031

Uncertainty quantification, and therefore sensitivity analysis, is a critical challenge in both validation and calibration. A lot has already been written on this topic in the computational literature and so we mainly discuss three highly speculative issues that are atypical of previously published themes. First, we discuss a formalization of the concept of code credibility that results from the use of benchmarks in verification and validation (V&V). Credibility is intended to be an important consequence of V&V; and calibration for that matter. We raise, but do not answer, the question of how credibility might be quantified. However such quantification may be achieved, it will have uncertainty associated with it. Second, we discuss a specific area of overlap between validation and calibration that is centered on how to deal with uncertainty in the physical models implemented in a CS&E code. This is the topic of *calibration under uncertainty* (CUU). Our primary conclusion is that recent calibration research that mathematically confronts the presence of this model-form uncertainty in statistical calibration procedures is important and coupled to validation issues. We speculate on the nature of this coupling, in particular that validation provides important information to calibration accounting for model-form uncertainty. We further argue that a full exploration of this issue might lead to the investigation of abstract learning theory as a quantitative tool in validation and calibration research. Finally, we speculate that uncertainty quantification has a role in verification. The use of uncertainty quantification in verification is probably not controversial, for example in statistical testing procedures, but our belief that the results of verification studies have uncertainty that requires quantification is. We explain this issue, but do not attempt to resolve it in this paper.

It is perhaps unclear why we are presenting an entire paper that mainly speaks to the issue of separation of calibration and validation. After all, is not the overarching goal of computational science to improve the associated calculations for given applications? Therefore, is not it natural to perform calibration to achieve this purpose? We believe that it is dangerous to replace validation with calibration, and that validation provides information that is necessary to understand the ultimate limitations of calibration. This is especially true in certain cases for which high-consequence CS&E prediction is required. These cases represent significant challenges for the use of CS&E codes and inevitably increase the importance of precisely distinguishing between validation and calibration in support of these uses. Our approach in this paper to calibration and validation emphasizes a kind of logical ideal. We do not emphasize practical issues, but the interested reader can find practicalities discussed in many of our references. We do emphasize that "real validation" and "real calibration" can be argued to be somewhat removed from the formalism and logical separation we stress in this paper. Murky separation of validation and calibration in real CS&E problems highlights the need to have some kind of logical foundation for clearly understanding the interplay of these concepts, especially for high-consequence applications of CS&E.

Section 2 presents a discussion of definitions of the various concepts mentioned above. Section 2.2 provides an illustration of the key ideas of verification, validation, and calibration using a computational fluid dynamics example (virtually the only CS&E example in the paper). Formalism of the concepts is introduced in Section 2.3, including of the concept of a benchmark and its comparison with calculations through comparison functions, and a notional formalism for credibility. In Section 3, we review common ideas of calibration (Sections 3.1 and 3.2), introduce some current research themes that generalize these ideas when considering uncertainty in the models that must be calibrated (Section 3.3), and introduce the possibility that computational learning theory might have some interest to our problems (Section 3.4). Section 4 briefly touches upon the role of sensitivity analysis in our discussion. We primarily provide some references, discuss the early appearance of sensitivity analysis in validation, and briefly comment on the presence of sensitivity analysis in credibility measures. Section 5 concludes the paper. We have tried to provide a useful set of references.

We emphasize that this paper presents some research ideas that are in early stages and somewhat speculative, but that we feel offer promising potential paths forward in calibration and validation. We introduce enough formalism to add some precision to our presentation, but this formalism does not reduce the amount of speculation in our discussion. Nor is the formalism enlisted to, in some sense, solve a particular problem in this paper. We hope that future papers will perform this role.

2. Calibration and validation

2.1. Guiding definitions

In this paper our underlying emphasis is on CS&E supported by the US Department of Energy's Advanced Simulation and Computing program (ASC, formerly called ASCI). A description of this program is given in [1]. The CS&E software developed under the ASC program is centered on the large-scale parallel numerical solution of systems of nonlinear partial differential equations (PDEs) of great complexity. The software implementations that accomplish this are called computer codes, or simply *codes* (see below).

Within this context, the current formal definitions of V&V used by the ASC program are as follows:

- *Verification* (*ASC*) is the process of confirming that a computer code correctly implements the algorithms that were intended.
- *Validation* (*ASC*) is the process of confirming that the predictions of a code adequately represent measured physical phenomena.

These definitions have a heritage that reaches back to definitions of V&V originally formalized by the Defense Modeling and Simulation Office (DMSO) of the United States Department of Defense (DoD) [2,3]. From the perspective of the computational engineering and physics communities, the definition of verification by the DoD does not make it clear that the accuracy of the numerical solution to PDEs should be included in the definition. To clarify this issue, the Computational Fluid Dynamics Committee on Standards of the American Institute of Aeronautics and Astronautics (AIAA) proposed a slight modification to the DoD definition [4]:

- *Verification* (*AIAA*) is the process of determining that a model implementation accurately represents the developer's conceptual description of the model and the solution to the model.
- *Validation* (*AIAA*) is the process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model.

While the AIAA definitions of V&V are more general than the current ASC definitions, we believe the ASC definitions to be compatible with the AIAA definitions. Calibration is not defined by ASC in [1]. We will give a definition of this term below, but we find it convenient to present this definition after we have defined the term benchmark.

We are only concerned with computational solutions of systems of PDEs, but of course V&V is of interest for a much broader class of simulation applications, such as discrete-event simulations and agent-based simulations. PDE solutions, generated by finite difference, finite element, spectral, or other numerical methods, follow the process of (1) writing down PDEs and required initial and boundary conditions; (2) developing mathematical algorithms for the numerical solution of these PDEs; (3) implementing these algorithms in a body of software; (4) execution of the code on computers and (5) analysis of the results.

A *code* is the body of software that implements the solution algorithms in step (3) above. This is the meaning of the word "code" in the ASC definitions of V&V above. Such codes are typically designed to be general purpose, for example computational fluid dynamics codes. This means that there is great flexibility in the specification of initial and boundary conditions, and in the numerical solution details (in particular discretization) of the numerical solution.

A *calculation* is a fixed choice of the input of a code, that is of the initial and boundary conditions, all physics modeling specification parameters, and all numerical parameters for a particular code that produce computational results, and the resulting output of the execution of the code with this choice of input. One could include within the components of a calculation other factors that influence the execution of the code as well, such as the computing hardware the calculation was performed on and the choice of post-processing software used to analyze the results of the calculation. We will not concern ourselves with these additional factors.

A CS&E *prediction* is simply a calculation that predicts a number or quantity or a collection of these quantities prior to or in lieu of their physical measurement. We have chosen the definition of prediction to be *anticipation of measurements* because that is the main emphasis for prediction in CS&E. With a computational prediction we are making a statement about physical phenomena without recourse to experimental observation of the specific phenomena *first*. Examples of such predictions include:

- Simulate an experiment without knowledge of its results or prior to its execution.
- Make scientific pronouncements about phenomena that cannot currently be studied experimentally.
- Use computation to extrapolate existing understanding into experimentally unexplored regimes.

We wish to distinguish our particular concerns here from the conventional notion of scientific exploration (where prediction may be wrong but still useful). We are interested in *confident prediction* using computation. In computational science, of course, to some degree confidence is correlated with belief in the quantitative numerical, mathematical and physical accuracy of a calculated prediction. Intuitively, confident prediction then implies having some level of confidence or belief in the quantitative accuracy of the prediction. This further implies a willingness to use the prediction in some meaningful way, for example in a decision process.

By introducing the expectation of accuracy in prediction as a foundation for confidence into this discussion we have therefore introduced the requirement for one or more measurement principles that we can use to quantify this accuracy. A *benchmark* is a choice of information that is believed to be accurate or true for use in verification, validation or calibration (defined below), one or more methods of comparing this information with computational results, and logical procedures for drawing conclusions from these comparisons. The analytic mathematical solution of a test problem would be true for use as a verification benchmark. In validation, accuracy of an experimental benchmark could be interpreted as small (enough) experimental error bars. Here we emphasize that the application of benchmarks performed in V&V is the primary basis for establishing our confidence that a prediction is accurate and useful. Clearly the choice of benchmark cannot be decoupled from the intended purpose of the benchmark. In the remainder of the paper we will sometimes use the word benchmark primarily in reference to the information defined by the benchmark, with the method(s) of comparison with calculations and drawing conclusions only implicit.

In principle, the accuracy assessment underlying confidence in a computational prediction can be performed through V&V either before or after the creation of the prediction. As we will argue in greater detail in Section 2, we believe that V&V should logically occur prior to the creation of a prediction. Computational prediction defined as performing calculations and reporting results does not require prior benchmarking, there is little or no confidence in the resulting prediction and this is not good practice. For example, a computational prediction regarding an upcoming experiment can be made with no attention to the problem of computational accuracy at all. Once the experiment is performed, merely comparing the experiment with the calculation does not create a benchmark. It is an independent thought process to decide what role the now-performed experiment might have as a benchmark, and revolves around questions about the relevancy and accuracy of the experiment.

Our confidence in the process of *accurate* prediction does require benchmarks. In particular, our judgment of the accuracy of a given prediction depends on our quantitative understanding of past computational performance of the code for other problems. Our concept of benchmark here is designed to capture the core of this past experience. In essentially all circumstances of practice in CS&E with complex codes, our current belief in the accuracy of a real prediction rests on a complex set of existing knowledge. Our concept of benchmark can be viewed as a concentration point of this knowledge. Examples of benchmarks include analytic test problems or the experimental information used in previous experimental–computational comparisons with the same code. Subjective expert opinion may also be deemed to be a benchmark. For example, this is often the basis for declaring one code to be a benchmark in a code-comparison activity [5].

Benchmarks are especially helpful for us in more completely quantifying our understanding of verification, validation and calibration. We quoted the ASC definitions of V&V above. A more technical discussion of the definitions of V&V is given by Oberkampf and Trucano [6]; see also Roache [7] for an alternative discussion. These references imply that there are myriad technical meanings associated with these terms, especially depending on the technical community that is engaged in the discussion. From a broader perspective, there is even debate as to whether a term like validation can be sensible when applied to computational models [8]. This debate tends to center on definitions of validation that imply some philosophical absolutes such as "correctness of physics." At its core in such a discussion, model validation means establishing that a model (or code) is true. When stated in this way, it is easy to dispute whether this is possible or not [9] although some [7] view this line of argument as outrageously irrelevant.

The concept of benchmarks helps us clarify the content of V&V and formalize the essence of the tasks that are performed in pragmatic circumstances. Let us first consider verification. From the software engineering perspective, the ASC definition of verification is compatible with the process of determining that requirements are correctly implemented in a code. For CS&E, Roache has stated that this effectively means that the equations implemented in the code are correctly numerically solved for the intended application. This is an important way of viewing the problem of verification because it emphasizes that verification is primarily a mathematical problem, as long as the issues specific to software implementation are acknowledged. We also have some expectation of generalization of the findings of verification tasks. If the equations are accurately numerically solved in one case, this is likely true for a broader set of cases given the mathematical nature of this problem. How broad this larger set of cases can be is an important question in particular subject matter applications.

The problem of verification in CS&E codes boils down to the following actions. Given a set of equations, (1) are the chosen solution algorithms mathematically correct? (2) is the software implementing these algorithms correct? (3) do particular choices of code input parameters yield accurate calculations when executed? This latter question emphasizes that a perfect code implementation of a perfect solution algorithm can yield an inaccurate calculation because of poor code input specifications, including an inadequately (for needed accuracy) defined computational mesh. This is not a trivial objection. For years, many important computational predictions have not been performed as accurately as needed because of restrictions on the characteristics of the computational mesh due to computing hardware limitations. There is no end in sight to this particular problem.

Benchmarks play an important role in performing verification, especially in assessing the answers to questions (1) and (2) above, because these questions cannot be completely answered by rigorous mathematical theorems in complex calculations. Oberkampf, Trucano, and Hirsch [6,10] discuss this issue in detail. The use of benchmarks in verification is called *testing* and has a major software engineering literature associated with it [11,12]. From our point of view, benchmarks also provide major input into the assessment of question (3), in the sense of aggregating and systematically integrating past accuracy experience to judge the computational accuracy of a current prediction. As noted by Oberkampf and Trucano [6], question (3) in principle can be answered well enough through a posteriori error estimation, *when and if* that technology achieves a sufficient level of rigor and generality to apply to the predictions CS&E must make in the future. While some believe that a posteriori error estimation already is sufficiently capable of addressing (3) for complex problems, we disagree. The other mathematical alternative of empirical convergence studies, even if it can be performed, ultimately has limited mathematical rigor associated with it. The use of benchmarks therefore remains a critical contributor to the resolution of question (3) in complex calculations. More generally, while we can honestly

characterize verification as a mathematical problem, it is highly unlikely that it will be fully addressed for realistic CS&E codes by convincing mathematical demonstrations. Instead, the variety of information available at any given time for attacking verification must include benchmarks and an understanding of their impact.

Validation deals with the question of whether the implemented equations in a code are correct for an intended application of the code. Again, from the software engineering perspective, the ASC definition of validation is the process of determining whether the requirements implemented in the code are correct for intended applications. For CS&E "correct requirements" means correct science and engineering. Thus, as Roache has emphasized, validation is a problem of physics, engineering, chemistry, and so on, not of mathematics. To avoid the philosophical debate hinted at above for validation, it is convenient for us to emphasize that in CS&E, validation is the process of quantifying the physical fidelity and credibility of a code for particular predictive applications through the comparison with defined sets of physical benchmarks. These benchmarks define what we will call the *validation domain* [13]. We assume in this paper that validation benchmarks are *always experimental*, for example as defined through dedicated experimental validation experiments. Predictive application of the code must therefore be interpreted as an interpolation or extrapolation beyond the validation domain.

Calibration is the process of improving the agreement of a code calculation or set of code calculations with respect to a chosen set of benchmarks through the adjustment of parameters implemented in the code. We emphasize that calibration and validation can differ simply through the choice of benchmarks. A benchmark set can be chosen specifically to facilitate the act of calibration and need not be relevant to validation, or verification for that matter. Clearly, the specific way benchmarks are selected and used to perform calibration is a technical and methodological issue, of which we will have more to say below. We expect this process to be highly subject matter dependent. In Section 2.3 we introduce formalism that allows us to explain more precisely the role of code input parameters in this discussion.

2.2. An illustration

Chen and Trucano [14] document a planar, inviscid two-dimensional validation study of strong shock reflection, where simulation was performed by the ALEGRA code solving the compressible Euler equations. The problem of interest is illustrated in Fig. 1, where the geometry of a typical calculation and a color representation of the computed two-dimensional density field at a given time are shown. The physical problem is that of an incident shock wave of given Mach number obliquely reflecting from a metallic wedge. In Fig. 1, the direction of the incident shock wave is given by the block arrow; the direction of the reflected shock wave is parallel to the wedge boundary, depicted by the second arrow. A double Mach reflection (DMR) simulation is shown in this figure. The shock reflection process transforms from regular reflection (RR) to single Mach or DMR dependent upon wedge angle θ and the incident shock Mach number. The details are unimportant for our discussion and can be found in Chen and Trucano. Information about the ALEGRA shock wave physics code can also be found in that reference.

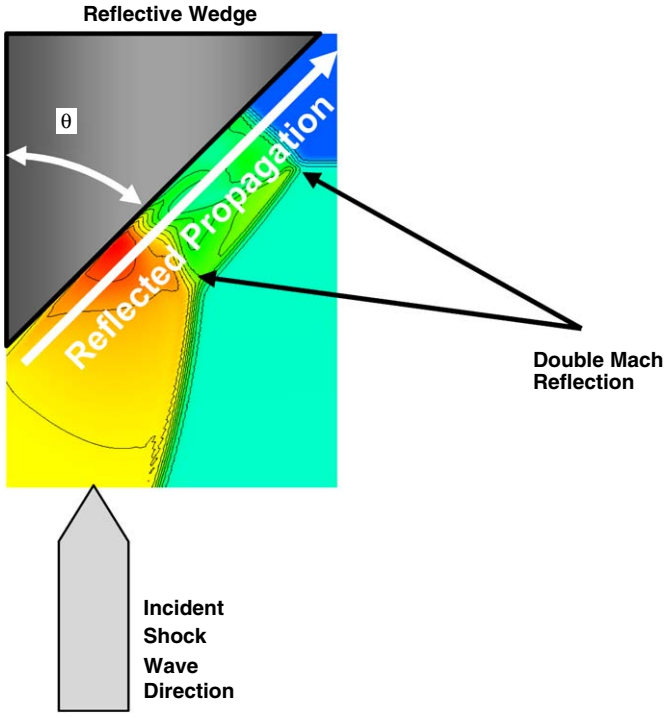

Fig. 1. Qualitative view of a calculation of double Mach reflection as presented in [14].

V&V can be illustrated in terms of these kinds of calculations. Fig. 2 demonstrates numerical accuracy concerns associated with verification by overlaying three separate resolution calculations reported in Chen and Trucano of the computed pressure as a function of distance parallel to the boundary of the wedge for conditions where DMR was observed experimentally. Clear differences are seen in these three calculations, most notably that a structure identifiable with DMR (two distinct shock waves) only begins to appear at the finest reported resolution. A posteriori error estimation techniques are not available in this study.

Our ability to draw conclusions from experimental data that may be compared with calculations like that in Fig. 1 is dependent upon the numerical quality of the calculations and the resolution at which experimental results are recorded. Verification assesses the numerical quality of the calculations, and has two components—*calculation verification* and *code verification* [7]. Fig. 2 is one means of studying numerical quality. The questions that should be

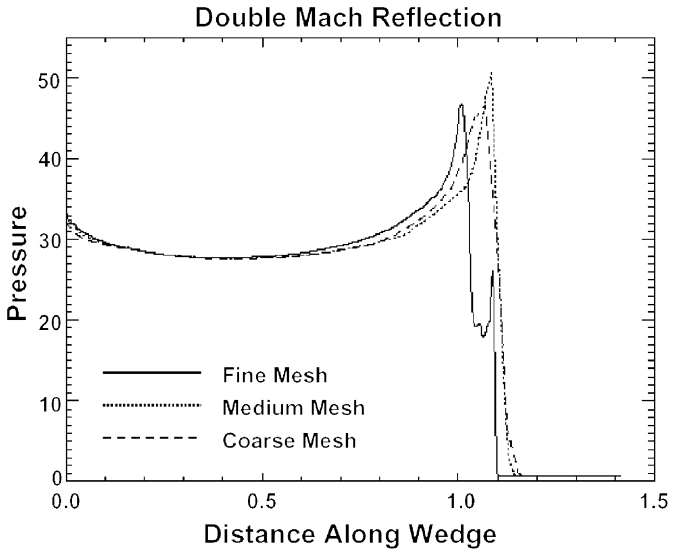

Fig. 2. Example of a convergence study performed in [14].

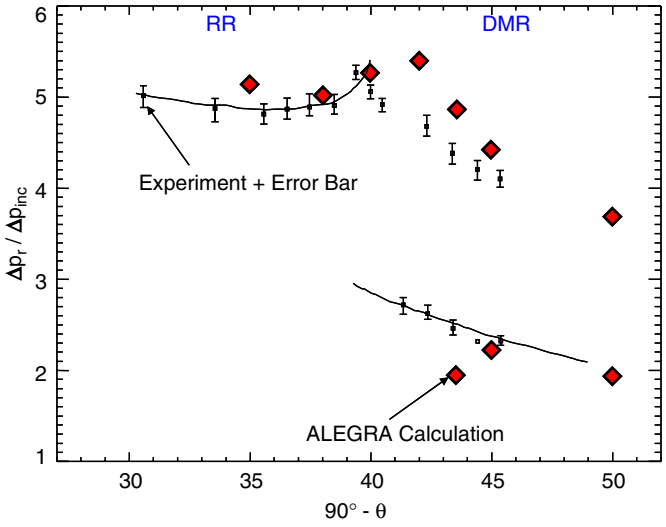

Fig. 3. Example of a comparison of calculated and experimental data [14].

asked, and that are simply implicit in Fig. 2, are many. Under calculation verification we may ask: What is the numerical error of a given calculation for a specified mesh and choice of input? How does this error depend on the mesh and computational parameters, such as numerical viscosities required for shock fitting in this case? More specifically, are calculations converged? That is, is the computed error below a specified tolerance? Under code verification, we might ask whether calculations that are believed to be converged, as confirmed perhaps through a careful grid convergence study, are in fact converged to the mathematically correct solution. This latter point is a question about the correctness of the solution algorithms and their software implementation. Therefore, its answer also requires rigorous appraisal of the absence of software bugs. This problem is made difficult and important by the fact that CS&E codes that compile and run and even produce results similar qualitatively to those in Figs. 1 and 2 may still have software bugs that destroy the credibility of the results.

The concept of validation is illustrated in Fig. 3, where experimental pressure data are compared with ALEGRA simulations. This figure is a variation of a figure first published by Sandoval [15], and overlays experimental normalized pressure data and specific ALEGRA calculations described by Chen and Trucano as a function of wedge shock angle. The vertical axis in this figure is defined by the normalized pressure ratio:

$$\frac{\Delta p_{\mathrm{r}}}{\Delta p_{\mathrm{inc}}} \equiv \frac{\text{Reflected Pressure} - \text{Ambient Pressure}}{\text{Incident Pressure} - \text{Ambient Pressure}},$$

Experimental data in the figure show the transition from RR to DMR near a wedge angle of 50°. The experimental data have error bars but, as explained in Chen and Trucano, there is insufficient information available to sharply characterize the statistical meaning of them. The experimental data in Fig. 3 define a validation benchmark, or several such benchmarks if one wishes to look at each experimental data point individually. One of the main purposes of the published Chen and Trucano study was to assess the *qualitative* ability of ALEGRA to approximate the transition from regular to DMR via comparisons with this benchmark. In principle validation should not even be attempted until the verification questions we stated above can be satisfactorily answered. In practice, unfortunately, for complex computational codes and models full resolution of these questions is virtually impossible. The conclusions in Chen and Trucano reflect this observation—the state of verification was so unsatisfactory (in particular, the lack of evidence of convergence of the calculations) that the authors could only draw qualitative conclusions from the comparison with the experimental data. There is clearly quantitative disagreement between the reported ALEGRA calculations and the experimental data, although the computed transition to DMR is in qualitative agreement with the experimental data. Until better understanding of the numerical error in the calculations can be achieved, the meaning of the quantitative disagreement is difficult to extract.

If we assume, nonetheless, that there is an acceptable verification basis for performing validation through comparisons with experiment and calculation as in Fig. 3 and drawing conclusions we are led to a variety of further validation-specific questions. For example, what do the experimental error bars in Fig. 3 mean? What are equivalent numerical error bars? Validation requires both types of error bars; what does it mean if one or both are missing? Are the depicted comparisons good, bad, or indifferent? What is the context of this evaluation? Why was this means chosen to compare experiment and calculation? Why are these experimental data considered to be benchmarks? Is there something better? Is the means

of comparison itself subject to sensitivities that should be quantified? Why was this particular physical problem, Mach reflection, chosen as a validation benchmark to begin with? What previous knowledge or context does this choice rest on? Where should we go next? What is the next benchmark? Or is this comparison enough?

A situation like Fig. 3 demonstrates the potential confusion of *calibration* with *validation*. In other words, a decision might be made to adjust certain parameters in the calculations to improve the agreement of ALEGRA (or any code undergoing this kind of comparison) with the presented experimental benchmark(s). A common example of this is the choice of a mesh when a calculation is otherwise not converged. One might apply mesh tuning and expert judgment about local refinement of the computational mesh to achieve some kind of better agreement with the presented data. This may then be accompanied by a statement of sufficiency of the presented computational accuracy based on this procedure. This process is *not validation*. It does not address any of the questions we posed above regarding validation. This is a *calibration process* and therefore has questionable explanatory or predictive content. Such a calibration also rests upon questionable mathematical grounds in the present example of calculations with uncertain mathematical accuracy. Neither does such a calibration address *verification*; the question of correctness of solution algorithms, their bug-free software implementation, and the particular mathematical accuracy of numerical solution of equations is not answered by comparison with *experimental* data, nor by tuning the computed agreement with experimental data.

The computations in Fig. 3 were those corresponding to the finest choice of mesh suggested in Fig. 2. However, this was a passive choice, not a deliberate calibration (attempt to better match the experimental benchmark). Chen and Trucano simply used the finest resolution calculations they could complete at the time in comparisons with experimental data. Fig. 2 provided evidence that finest resolution calculations were important because the physical structure of greatest interest, DMR, only emerged at the finest resolution. But this choice does not resolve the underlying questions of verification because only incomplete convergence studies are reported and, therefore, the mathematical accuracy of these calculations remains an open question. The numerical error contribution that the particular meshing may make to the agreement or lack of agreement with the data in Fig. 3 remains open, and that is why numerical error bars were not reported in Chen and Trucano.

We also comment that a very simple example of another common kind of calibration in a compressible flow problem such as that shown in Fig. 1 is to adjust the equation of state of the gas defined in the numerical calculations in an attempt to better match experimental benchmarks as in Fig. 3. This may or may not be possible; and it may or may not be useful. We emphasize here that this procedure is still logically separate from verification, hence still leaves that question open. The value of such a calibration ultimately rests upon sufficient verification. If numerical errors are large or if code bugs are present in Fig. 3, what is the point of adjusting a physical parameter in the calculation to achieve better agreement with the benchmark?

This simple example emphasizes that verification questions must really be answered for calibration as well as validation. Our comments above argue that this may be quite difficult. But in the absence of adequate verification, the results of neither validation nor calibration can really be fully trusted. If one insists on considering certain kinds of calibration to be compensation for inadequate numerical resolution the foundation provided by verification is removed in the presented computational results, making consequential validation assessments that much harder, if not rendering them useless. We reasonably ask what the rationale is for calibrating a code to compensate for perceived numerical discretization inadequacy when the root of the problem might be code bugs instead. For instance, when may such a calibration successfully match superficial quantitative features in the chosen benchmark(s) but obscure or ignore important qualitative information that has been corrupted because of algorithm or implementation errors?

We insist that calibration and validation are distinct. While validation should provide a better foundation for calibration and guide its use, calibration does not replace validation, nor does it replace verification. To the extent that the use of calibration confuses or hides the fact that V&V are poor, confidence in the credibility of the resulting calculations is misunderstood. For applications to high-consequence problems, such as governmental policy issues, we believe that this is undesirable. In the circumstances of poor V&V, calibration should be used cautiously and with a certain degree of pessimism about its ultimate value.

2.3. Some formalism

2.3.1. Alignment of calculations and benchmarks

Whether we are performing verification, validation, or calibration, the overarching purpose of the task is to perform the right calculations for the task; to compare these calculations with the right benchmarks; and to draw the right conclusions. To enhance our discussion of these issues we now find it convenient to introduce enough formalism to sharpen the distinctions between these tasks. This will further clarify our position on validation versus calibration. The reader should view the following presentation as mainly notional. We provide sufficient references to specific instances of these formalisms to aid the reader who is interested in greater details.

We emphasize again that we are focused on computational codes that solve PDEs. We declared previously that the word calculation would be reserved for the execution of a computational code for specific input, producing a given output. Let $M(\cdot)$ denote the *code*, which is a nonlinear (in

general) mapping from the input to the output, thus $M(\cdot) : \boldsymbol{P} \rightarrow \boldsymbol{O}$. The input is written as *parameter vector* in the space $\boldsymbol{P}$, $\vec{p} \in \boldsymbol{P}$, where $\boldsymbol{P}$ is (typically) high-dimensional (for example, $\boldsymbol{P}$ is usually a subset of m-dimensional Euclidean space $\mathbb{R}^m$) and serves to uniquely define the input for a particular calculation. A calculation is then a specific choice of input $\vec{p}$, and the output of the code applied to this input, $M(\vec{p})$. Different choices of $\vec{p}$ generate different calculations. The output space $\boldsymbol{O}$ is, in principle, equally complex, reflecting the potential complexity of the output of real CS&E codes.

The input parameter vector $\vec{p}$ is necessary for specifying a unique input set for the underlying code and may have components that are often functions, such as a permeability field in porous flow, or probability distributions for stochastic parameters. Furthermore, the components of $\vec{p}$ will possibly be other vectors or matrices, in addition to scalars. The complexity of the input is considerable and highly subject-matter dependent. When we state that the vector $\vec{p}$ is high-dimensional we mean that the number of input parameters may be greater than the number of nominal calculations that might be performed during the course of a verification, validation, or calibration study in the absence of uncertainty quantification procedures. Potentially relevant $\vec{p}$'s form a small subset of the indicated space for most codes. Because of this, we find that a decomposition of this vector is important for our purposes. Thus, we introduce the notation

$$\vec{p} = (\vec{p}_{\mathrm{A}}, \vec{p}_{\mathrm{N}}) \in \boldsymbol{P}_{\mathrm{A}} \times \boldsymbol{P}_{\mathrm{N}}. \tag{1}$$

In Eq. (1), $\vec{p}_{\mathrm{A}}$ is the component of the parameter vector that specifies *alignment* with the intended model application. In other words, a specific calculation for an intended application is defined by a choice of $\vec{p}_{\mathrm{A}}$. Therefore, we *assume* that $\vec{p}_{\mathrm{A}}$ is sufficient to parameterize the application domain for a specified use of the code. The $\vec{p}_{\mathrm{A}}$ vector, for example, specifies geometry and other initial data, and boundary conditions. Whether the entire application domain defined this way is accessible through validation or calibration is another matter.

The vector $\vec{p}_{\mathrm{N}}$ specifies numerical parameters and other quantities that are necessary to execute a calculation and control its numerical accuracy. $\vec{p}_{\mathrm{N}}$ includes parameters necessary for adjusting the numerical influence of discretizations, including both temporal and spatial characteristics. $\vec{p}_{\mathrm{N}}$ can include iterative accuracy controls, computational parameters devoted to artificial smoothing and diffusion or other similar regularizations, and parameters that control automated algorithm operation, as in automatic mesh refinement control and hybrid algorithm operation. The vector $\vec{p}_{\mathrm{N}}$ may or may not be independent of $\vec{p}_{\mathrm{A}}$. For example, a change in $\vec{p}_{\mathrm{A}}$ might change numerical characteristics controlled by $\vec{p}_{\mathrm{N}}$.

For example, suppose that we are performing validation. In this case, the calculation is to be compared with experimental data. The model is constrained by the need to calculate in alignment with the chosen experimental benchmark to the highest degree. This means ideally that the computational geometries, materials, and other initial data should be the same as the experimental benchmark, and the boundary conditions should be the same. The parameter vector $\vec{p}_{\mathrm{A}}$ is the means by which this is accomplished, or approximated if it cannot be perfectly achieved.

If we consider the example in Figs. 1–3, the subject code is ALEGRA. The purpose in this case is to perform ALEGRA calculations that compare with oblique shock reflection experiments. The vector $\vec{p}_{\mathrm{A}}$ in the calculations of Fig. 3 has the following components:

ρ_0—Initial gas density;
P_0—Initial gas pressure;
I_0—Initial gas specific internal energy;
γ—Ideal gas gamma constant;
M—Incident shock wave Mach number;
θ—Wedge angle;

(In calculations, the wedge is treated as a boundary condition, not as a material included in the calculation.) These parameters are sufficient for alignment with the experiment because real gas effects (which would increase the number of parameters required to describe the gas) and roughness of the wedge (which would require parameters for an appropriate boundary condition or parameters involved in the treatment of a real wedge in the calculations) are considered to be irrelevant in the extraction and interpretation of the presented experimental data.

The vector $\vec{p}_{\mathrm{N}}$ in these calculations includes (but is not restricted to) artificial viscosity coefficients, hourglass control coefficients, meshing parameters, and choice of numerical algorithm features (such as remap characteristics) in the ALEGRA numerical solution of the Euler equations. The parameters in $\vec{p}_{\mathrm{N}}$ do not control the alignment of the validation calculations presented in Fig. 3 to the specific experiment performed for each data point. The only alignment parameter actually varied in Fig. 3 is the wedge angle. The initial data, including Mach number of the incident shock, did not vary. We are not accounting for uncontrolled variability in the initial data of the experiments, nor for inaccuracy in measuring experimental parameters, such as the wedge angle.

The alert reader will notice that, in fact, the calculations presented in Fig. 3 do not precisely align with the experimental benchmarks, because the choice of computational wedge angle was often different than that for the experimental data. This was a decision made in the original modeling approach, and does not affect our use of this example to illustrate the principle of alignment.

We have previously discussed the concept of alignment in [16]. We prefer to deal with the issue in a simpler form here, but in fact alignment poses a problem that is quite difficult for complex validation and calibration tasks. Our concept of alignment also resembles discussion of the role

of "cues" for general forecasting methods in the paper of Stewart and Lusk [17]. For example, in a perfect world the experimental benchmark that we may wish to use will be exactly defined by the parameter vector $\vec{p}_A$ (the experiment has neither more nor less nor different parameters than the calculation) and we know precisely what the values of this vector are for the chosen benchmark. One could choose to make this a necessary condition for applicability of an experiment as a validation benchmark.

Unfortunately, if one enforced that restriction, one would rarely be able to use experimental data as a benchmark. Even in the simpler case of verification, we may not be able to properly define $\vec{p}_A$ to align with the assumptions underlying a chosen analytic solution of the PDEs of interest, or with other mathematical benchmark requirements. It is well-known that there exist analytic solutions, for example in compressible hydrodynamics, that may require initial and boundary conditions that are not implemented in a given code. A code modification is required to allow full alignment with such a benchmark. The Method of Manufactured Solutions typically creates this problem through the need to create problem-specific boundary conditions or source terms for the derived test problems [7,18].

For validation or calibration based on experimental benchmarks, the choice of $\vec{p}_A$ for the code is often at best a poor approximation of the experimental reality. For example, there may be parameters of importance to the experiment that are not even recognized in $\vec{p}_A$. Attention to uncertainty in this specification is thus required. And, even if we know that $\vec{p}_A$ is a good approximation to an exact specification of an experimental benchmark, it is unlikely that its experimental values are known precisely. Hence the experimental specification of $\vec{p}_A$ contains uncertainty. These two issues are major contributors to the uncertainty underlying experimental benchmarks that we must deal with in validation and experimental calibration.

Technically, *uncertainty* has two distinct meanings (see [19–28]) that we use in this paper. *Aleatory* uncertainty, also called irreducible uncertainty, is random. Aleatory uncertainty is used to describe inherent variation associated with a quantity or phenomenon of interest. Further information does not usefully reduce the uncertainty. This is in contrast with *epistemic* uncertainty, also called irreducible uncertainty (and called subjective uncertainty by Helton), which reflects lack of knowledge about the quantity or phenomenon. The implication is that this uncertainty can be usefully reduced in principle through the acquisition of more information. While probability and statistics is the defined means of quantifying aleatory uncertainty, the quantification of epistemic uncertainty is more difficult. A recent issue of the journal Reliability Engineering and System Safety is entirely devoted to this topic [29].

Proper definition of $\vec{p}_A$ is an epistemic uncertainty problem, while particular experimental values of properly aligned elements of $\vec{p}_A$ is an aleatory uncertainty problem. In general, we do not expect to have all of "nature's parameters" available in the code's $\vec{p}_A$. However, one of the most important aspects of designing and performing dedicated validation experiments [13] is to maximize the likelihood that the code's $\vec{p}_A$ is sufficient to properly align code calculations with experiments. For validation, at least, we thus have some expectation of the sufficiency of $\vec{p}_A$ to achieve alignment with the experiment. Scientific discovery (as opposed to validation) is more likely to result in doubt about the ability of $\vec{p}_A$ to achieve alignment with an experiment, as well as the ability of $M(\cdot)$ to provide a rational basis for that alignment. Further discussion of these issues is beyond the scope of this paper. We will simply assume the existence of $\vec{p} = (\vec{p}_A, \vec{p}_N)$ and *ignore* in particular the potential uncertainty in the definition and values of $\vec{p}_A$.

2.3.2. Benchmarks

We introduced the decomposition of the parameter $\vec{p} = (\vec{p}_A, \vec{p}_N)$ for the purpose of refining the definition and application of benchmarks. We now state that a *benchmark* is a choice of $\vec{p}_A$ and the benchmark data associated with that choice (along with the implied methods of comparison with calculations and conclusion-drawing procedures). We write a benchmark as $B(\vec{p}_A)$. This notation suggests that a benchmark simply maps the alignment parameters into information, either experimental or mathematical (or computational, if an alternative code calculation is the chosen benchmark), that can be quantitatively compared with some or all of the output of the corresponding code calculation $M(\vec{p}_A, \vec{p}_N)$. $B(\vec{p}_A)$ may be extremely complex, for example including both temporally and spatially resolved data. Certainly, in the case of experimental data by simply writing the benchmark as a function of a parameter vector we are claiming more than will be the case in many experiments. However, as we are emphasizing the logical nature of benchmarks here, we will take this simplifying liberty.

In the case of experimental data, the benchmark information $B(\vec{p}_A)$ will have uncertainty in the form of bias and variability, for example due to diagnostic uncertainty, and experimental bias and variability. We may also have uncertainty about whether or not the experiment really provides required benchmark information, such as that specified for validation benchmarks by Trucano et al. [13]. One should most generally consider the benchmark information $B(\vec{p}_A)$ as a random variable, process or field. If we formally acknowledge aleatory uncertainty in $\vec{p}_A$, then $B(\vec{p}_A)$ is also a function of a multivariate random variable. This has technical implications for how we should ideally perform verification, validation, and calibration, as it will strongly influence the methods used to compare the benchmark information and code calculations, as well as inferential procedures dependent upon those comparisons.

To the degree that one feels uncertain with whether a given set of data should be used as a benchmark, say for

validation, one is likely dealing with epistemic (lack of knowledge) uncertainty. For example, if person A claims "It is important to match experimental data E", person B may alternatively argue "E is the wrong data," C may argue "But we do not know how close we need to be to E," and D may argue "The theory required to interpret the experiment is wrong." All of these arguments may contribute epistemic uncertainty to the benchmark. This issue is not academic, but must be dealt with constantly in experimental validation [13].

Different benchmarks defined as above can result from either varying the benchmark function $B(\cdot)$, $\{B_j(\cdot), j = 1,\ldots,J\}$, by varying the alignment parameter, $\{\vec{p}_{\mathrm{A},i}, i = 1,\ldots,n\}$, or both. An example of the former case is to perform measurements of different physical quantities at the same spatial location. An example of the latter is to measure the same physical quantity at different spatial locations or times. In the most general case, a set of benchmarks is then defined by $\{B_j(\vec{p}_{\mathrm{A},i}), i = 1,\ldots,n; j = 1,\ldots,J\}$. In the following, we will only consider variation of the alignment vector in defining a set of benchmarks. Thus, we define a set of benchmarks as the collection of benchmark data $\{B(\vec{p}_{\mathrm{A},i}), i = 1,\ldots,n\}$. We will also assume that the benchmark function changes for verification, validation and calibration, without introducing this explicitly in the notation.

Verification, validation and calibration are all focused on defining the appropriate set $\{B(\vec{p}_{\mathrm{A},i}), i = 1,\ldots,n\}$. We will loosely state that a chosen set of benchmarks defines the *verification* (*validation*, *calibration*) *domain* in the parameter space $\boldsymbol{P}_{\mathrm{A}}$ through the set $\{\vec{p}_{\mathrm{A},i}, i = 1,\ldots,n\}$. This domain is therefore a discrete set, although it is clearly desirable to think of it as lying within a continuous region that contains the discrete choices of $\vec{p}_{\mathrm{A}}$ as much as possible. To what extent this can be done is also an open issue, although this assumption is commonly made in practice. The reason for this is that results of comparisons of code calculations with such a set of benchmarks are often interpolated to infer presumed code credibility (or not) *between* the benchmarks. This implies an assumption about an overall domain containing the set $\{\vec{p}_{\mathrm{A},i}, i = 1,\ldots,n\}$. The reader should observe that we are enforcing our constraint of alignment by emphasizing that *only* the parameter vector $\vec{p}_{\mathrm{A}}$ is meaningful in defining these domains; $\vec{p}_{\mathrm{N}}$ does not enter this discussion. For example, in Fig. 3 the set of benchmarks is $\{B(\vec{p}_{\mathrm{A},i}) = [\Delta p_{\mathrm{r}}/\Delta p_{\mathrm{inc}}](\theta_i), i = 1,\ldots,18\}$ (there are 18 experimental data points in the figure, one of which has no reported error bar).

We are choosing our words in this discussion carefully. We do not care about the issue of whether experimental data are "true" in some absolute sense of the philosophy of science. We are simply defining a benchmark to be a choice of data that we are willing to use for purposes of verification, validation or calibration, that is, to provide information for some stated goal. We assume that it is understood that there are good reasons to use the specified information as a benchmark. We have the expectation that these reasons will include evidence that the benchmark is accurate and that a calculation can be properly aligned with it. We also have the additional expectation that if there is little or no evidence that a benchmark is actually accurate or believable for the purposes of the stated task, then it will not be used.

Clearly some principles should guide or strengthen the logic of accepting a benchmark for the purpose of verification, validation, or calibration. We have defined such a set of principles specifically for validation [13]. We have also documented arguments as to why we believe that code calculations themselves are undesirable benchmarks [5]. The debate, of course, is centered on the use of words like "reasonable" and "evidence." For example, it is fair to ask whether or not we believe that the data shown in Fig. 3 are a "reasonable" validation benchmark. This issue was discussed in the original reference [14], and it was determined that in fact there are significant problems associated with use of these data in a precise validation exercise.

The fundamental purpose of benchmarks is to draw specific conclusions from their comparison with calculations. In the case of *verification*, this purpose is to assess the *mathematical accuracy* of the numerical solutions. For *validation*, this purpose is to assess the *physical fidelity* for a stated application of the mathematical equations solved in the code. For *calibration*, the purpose is to choose parameter values that improve the agreement of the code calculations with the chosen benchmarks, in the belief that such tuned accuracy improvement will increase the believed credibility of the code (a goal we consider to be incorrect, as we commented in the Introduction). The choice of benchmarks must vary depending on the purpose of the comparisons.

Uncertainty is present in the typical code calculations we are interested in, not only in the benchmarks themselves. Uncertainties in calculations have been discussed in many places (see, for example [30]). Uncertainty in a calculation $M(\vec{p})$ may be present due to uncertainty in $\vec{p}_{\mathrm{A}}$ or uncertainty in the specification of $\vec{p}_{\mathrm{N}}$, although the latter point is more controversial. However, uncertainty in $M(\vec{p})$ also arises independent of $\vec{p}$ for two reasons. First, there is the epistemic uncertainty called structural uncertainty, or "model form" uncertainty. Its origin is the question of whether the so-called "conceptual model" [6] underlying the mathematical equations solved in the code is appropriate for the intended use. This includes unknown–unknown factors, such as what physics is missing from the conceptual model that is necessary to describe the relevant physical phenomena. To the degree that validation is incomplete this uncertainty looms large. (Of course, model-form uncertainty poses a broader problem than simply the issue of its contribution to uncertainty in code calculations.) Second, there is the epistemic uncertainty associated with imprecise characterization of the computational error in any specific

calculation. To the degree that verification is incomplete this uncertainty looms large.

Thus, to perform the right comparison between a code calculation and a benchmark, in practice as well as in principle the uncertainty present in both should be *acknowledged and quantified* in that comparison. This increases the difficulty of verification, validation and calibration considerably. The potential presence of epistemic uncertainty in both the benchmark and the code makes the task even harder. The reader should now assume that all uncertainties under scrutiny in this paper are quantifiable via probability. We thus avoid additional philosophical arguments about how to properly represent epistemic uncertainties. The interested reader should consult Ref. [29].

2.3.3. Comparisons

We now further formalize comparisons between calculations and benchmarks. Up to this point, we have treated the comparison dimension of benchmarks as implicit. Here we explicitly discuss it. Given the benchmark $B(\vec{p}_{\mathrm{A}})$ and a corresponding aligned calculation $M(\vec{p}_{\mathrm{A}},\vec{p}_{\mathrm{N}})$ a *comparison* is then a non-negative function of the two that is a measure of how similar (or different) they are. We write this as $\boldsymbol{D}[M(\vec{p}_{\mathrm{A}},\vec{p}_{\mathrm{N}}),B(\vec{p}_{\mathrm{A}})]$. For example (as further observed below) the comparison could simply be a norm (in an appropriate space) of the difference of $M(\vec{p}_{\mathrm{A}},\vec{p}_{\mathrm{N}})$ and $B(\vec{p}_{\mathrm{A}})$, in which case the comparison is a functional that is equal to zero when the benchmark and calculation are identical (a highly unlikely possibility). Given the complexity and uncertainty that characterizes both the benchmark and the calculation, however, we reserve the possibility that the comparison cannot be expressed simply as the simple difference of the two quantities.

The concept of a comparison can be extended to a family of benchmarks. For example, supposed the benchmark family is simply parameterized by a set of alignment vectors, as we have assumed here. Then, given the family of benchmarks $\{B(\vec{p}_{\mathrm{A},i}), i=1,\ldots,n\}$ and the aligned family of calculations $\{M(\vec{p}_{\mathrm{A},i},\vec{p}_{\mathrm{N},i}), i=1,\ldots,n\}$ the resulting set of comparisons is a set of non-negative real numbers:

$$\{\boldsymbol{D}^i \equiv \boldsymbol{D}[M(\vec{p}_{\mathrm{A},i},\vec{p}_{\mathrm{N},i}),B(\vec{p}_{\mathrm{A},i})],\ i=1,\ldots,n\}. \tag{2}$$

For the more general case of the benchmark function $B(\cdot)$ itself changing, rather than just the alignment vector, then it is possible that the comparison will also change. We point out below that we expect this change of comparison function to be inevitable in distinguishing verification, validation, and calibration, but we don't want to wrestle with this additional complexity.

The main logic underpinning (2) is the simple idea that if all $\boldsymbol{D}^i[\cdot,\cdot]$ are "small" real numbers for the presented set of benchmarks, then the calculation and benchmarks are "close" over a subset $\{\vec{p}_{\mathrm{A},i}, i=1,\ldots,n\}$ of the alignment parameter space, and this should lead to a conclusion about, say, verification or validation of the model. What "small," "close" and "conclusion" mean in this logic, is in almost all cases dominated by subject matter and the specifics of the chosen benchmarks and comparisons. An explicit specification of these words for a defined set of benchmarks would complete a full formalism of a benchmark set that would be compatible with our original definition of the term benchmark, but this is a task that we do not attempt in this paper. Such a formalism would eliminate, for example, intuitive statements of how good the agreement of a calculation with experimental benchmark information is, and replace it with quantification of the comparison and requirements on the precision of the comparison that allow the words "good agreement" to be used to describe it.

In many cases the specific choice of $\boldsymbol{D}[\cdot,\cdot]$ is literally a mathematical norm, such as absolute value in the case of scalars (as in Fig. 3), or l^p norms for more complex data. In the case of benchmark data like that in Fig. 3, one example of a comparison is

$$\boldsymbol{D}[M(\vec{p}_{\mathrm{A},i},\vec{p}_{\mathrm{N},i}),B(\vec{p}_{\mathrm{A},i})] = \left|\frac{\Delta p_{\mathrm{r}}}{\Delta p_{\mathrm{inc}}}(\theta_i)^{\mathrm{calc}} - \frac{\Delta p_{\mathrm{r}}}{\Delta p_{\mathrm{inc}}}(\theta_i)^{\mathrm{expt}}\right|. \tag{3}$$

Eq. (3) is written as if the calculations and experimental benchmarks in Fig. 3 had been defined at the same values of θ. Since this is not precisely true for the particular data of Fig. 3, in that case an alternative comparison can be defined in which we interpolate between the experimental benchmarks to determine a value of $[\Delta p_{\mathrm{r}}/\Delta p_{\mathrm{inc}}](\hat{\theta}_i)^{\mathrm{expt}}$ for the angles $\hat{\theta}$ that was actually used in the calculations and then apply the same absolute value comparison as in (3). This procedure is permissible if the interpolated experimental data is judged to still be a benchmark, which need not be the case. (Clearly, the desirable procedure for this example would have been to precisely align the original calculations with the chosen benchmarks.)

Appropriate acknowledgement and quantification of uncertainty in the benchmarks and calculations in (2) adds significantly greater technical difficulty to the comparison exercise, as well as to the understanding of "close" and "conclusion" previously mentioned. For example, whatever the set of calculated outputs $\{M(\vec{p}_{\mathrm{A},i},\vec{p}_{\mathrm{N},i}), i=1,\ldots,n\}$ may mean from the deterministic point of view, it means something different if we have quantified computational uncertainty in the stated set of calculations. At the very least, we would then hope that we have enough information to replace $\{M(\vec{p}_{\mathrm{A},i},\vec{p}_{\mathrm{N},i}), i=1,\ldots,n\}$ with a collection like $\{M_{\mu,i}, M_{\sigma,i}\}, i=1,\ldots,n$, where $M_{\mu,i}$ is the mean of the calculation (calculation now interpreted as a random variable, process, or field) at $(\vec{p}_{\mathrm{A},i},\vec{p}_{\mathrm{N},i})$ and $M_{\sigma,i}$ is a measure of fluctuation of the calculation at $(\vec{p}_{\mathrm{A},i},\vec{p}_{\mathrm{N},i})$, perhaps chosen as a statistic of central tendency such as variance. Separate comparisons would examine differences between the mean and variance of benchmarks and calculations in this case. In general, probability distribution differences between benchmarks and calculations could be compared. This would require a more elaborate formalism.

Conclusions based on uncertainty quantification of the information in the benchmarks, the calculations, or both must of necessity be interpreted as statistical or probabilistic information. For random variables with known probability distributions that appear in the comparison (2), the quantification $\boldsymbol{D}[\cdot,\cdot]$ can be achieved by comparing probability distributions or their moments. Greater complexity results when only statistical characteristics are known. In such a case, probability distributions are now estimated, leading to more complicated $\boldsymbol{D}[\cdot,\cdot]$'s, such as might involve the use of estimated probability distribution parameters and confidence intervals (CI) [31]. If epistemic uncertainty is also of concern, then combined statistics from families of probability distributions may be compared (through the use of second-order probability, for example). We will not probe more deeply into this challenge, except to emphasize that it exists. Helton [24] discusses this issue from a general perspective; other references of interest are [27,28].

The representation (2) of a set of calculation–benchmark comparisons presumes ideal parameter alignment; that is, the choice of $\vec{p}_{\mathrm{A},i}$ is the same for each benchmark and calculation being compared. This leads to questions specific to uncertainty in the alignment vector itself. If we had to apply (2) with a significantly different choice of $\vec{p}_{\mathrm{A}}$ in a benchmark versus that in a calculation, it is unlikely that (2) would be small. Furthermore, even if the comparison was small in such a case, what does that mean? But, what does "significantly different" really mean? The fundamental issue here is that if uncertainty is acknowledged in the definition of $\vec{p}_{\mathrm{A}}$, then the value of the alignment vector in the benchmark need not exactly agree with that in the calculation. For example, if we ignore the epistemic uncertainty contribution mentioned above and only consider variability in $\vec{p}_{\mathrm{A}}$, then specific choices of $\vec{p}_{\mathrm{A}}$ in (2) should be viewed as samples from an underlying probability distribution. Do these sample values need to be identical for the benchmark and the calculation to apply the comparison (2)? In principle, both the calculation and benchmark components in (2) should treat $\vec{p}_{\mathrm{A}}$ as a random quantity, but in practice this is often not done. For example, if an experimental benchmark is defined so that $\vec{p}_{\mathrm{A}}$ contains an experimental parameter with a probability distribution, often the value of this parameter used in a calculation is the mean of this distribution. Is this the most useful or appropriate way to conduct a calculation–benchmark comparison? Including epistemic uncertainty in $\vec{p}_{\mathrm{A}}$ makes the problem more difficult. Here, we simply intend to acknowledge the presence of this difficulty without further analysis. We emphasize that our view of alignment between benchmarks and calculations in comparisons like (2) serves as a logical guide, but its practical application is not transparent.

In principle, the collection of comparisons in (2) can be summarized in a single non-negative real number (which may or may not be desirable). A summary comparison in this case is written as the sum of the individual comparisons:

$$\begin{aligned}\boldsymbol{D}\{M(\vec{p}_{\mathrm{A},i},\vec{p}_{\mathrm{N},i}),B(\vec{p}_{\mathrm{A},i})i=1,\ldots,n\}&=\sum_{i=1}^{n}\boldsymbol{D}^{i}\\&=\sum_{i=1}^{n}\boldsymbol{D}[M(\vec{p}_{\mathrm{A},i},\vec{p}_{\mathrm{N},i}),B(\vec{p}_{\mathrm{A},i})].\end{aligned}\tag{4}$$

For example, if we assume that we had calculations ideally aligned with each of the 18 experimental data points in Fig. 3, then Eq. (4) would look like:

$$\boldsymbol{D}\{M(\vec{p}_{\mathrm{A},i},\vec{p}_{\mathrm{N},i}),B(\vec{p}_{\mathrm{A},i})i=1,\ldots,18\}=\sum_{i=1}^{18}\left|\frac{\Delta p_{\mathrm{r}}}{\Delta p_{\mathrm{inc}}}(\theta_i)^{\mathrm{calc}}-\frac{\Delta p_{\mathrm{r}}}{\Delta p_{\mathrm{inc}}}(\theta_i)^{\mathrm{expt}}\right|.$$

The summary (4) also suggests the possibility of weighting of individual contributors to a collective benchmark, for example as in the form

$$\boldsymbol{D}\{M(\vec{p}_{\mathrm{A},i},\vec{p}_{\mathrm{N},i}),B(\vec{p}_{\mathrm{A},i})i=1,\ldots,n\}=\sum_{i=1}^{n}w_i\boldsymbol{D}^{i},\qquad\sum w_i=1,\quad w_i\geqslant 0.\tag{5}$$

In terms of Fig. 3, Eq. (5) expresses the possibility that we may care about certain wedge angles in the figure more than others for purposes of validation. (In the original Chen and Trucano reference, this was exactly the case, although not expressed in this formal sense. In that study, the transition to DMR near a wedge angle of 50° in the benchmark data was of the greatest interest for comparison with calculations.)

Allowing different choices of benchmarks, rather than only different alignment vectors $\vec{p}_{\mathrm{A}}$, means that quite heterogeneous comparisons might be lumped together within equations like (4) or (5). For example, one of the experiments in Fig. 3 could have had a time-resolved measure of a gas field variable at a spatial location, so that an appropriate comparison for that data would be a choice of a function space norm. In general, $M(\cdot)$ and $B(\cdot)$ have widely different output information, with correspondingly different units (for example, energy versus pressure). This means that summary weighted comparisons in the form of (5), if desirable, require properly scaled and dimensionless individual comparisons.

Because there is no basis for expecting verification, validation or calibration benchmarks to be the same, there is also no reason to expect that verification, validation or calibration comparisons should be the same. Thus, for the same code calculation $M(\vec{p}_{\mathrm{A}},\vec{p}_{\mathrm{N}})$, we may have different choices of comparisons $\boldsymbol{D}_{\mathrm{VER}}[\cdot,\cdot]$, $\boldsymbol{D}_{\mathrm{VAL}}[\cdot,\cdot]$, and $\boldsymbol{D}_{\mathrm{CAL}}[\cdot,\cdot]$ defined. The choice of a collection of benchmarks $\{B(\vec{p}_{\mathrm{A},i}),i=1,\ldots,n\}$ can also vary depending on the purpose of the exercise. As emphasized above, comparisons should acknowledge uncertainty in the calculations and the benchmarks in general. In calibration, uncertainty is traditionally acknowledged in the benchmark data, but

not in the associated calculations. (We discuss this in detail in Section 3.) In validation, the quantitative treatment of uncertainty in both benchmarks and in calculations is understood to be a major challenge. Oberkampf and Trucano [6] present an analysis of useful general characteristics of a validation comparison (called a metric in that paper) and a specific example. A different formulation of validation comparisons is found in Zhang and Mahadevan [32] and Mahadevan and Rebba [33].

In verification, we believe that the role of uncertainty has not been recognized to the degree that it should be. In our view, uncertainty in verification arises from the need to report calculation accuracy rigorously, and the dependence of any reported accuracy upon mathematical rigor of the algorithms in the code, as well as their software implementation. Reliability of software as a probabilistic concept is recognized [34] and provides some foundation for addressing this topic in the breadth we are interested in, but discussion of this concept is beyond the scope of this paper (although we do make a further comment about this topic in Section 2.3.4).

V&V have the goal of quantitative assessment of the code credibility for a specified application. In terms of our discussion in this paper, we emphasize that V&V at least require (1) the choice of the appropriate benchmarks $\{B(\vec{p}_{\mathrm{A},i}), i = 1,\ldots,n\}$; (2) the evaluation of $\boldsymbol{D}_{\mathrm{VER}}[\cdot,\cdot]$ and $\boldsymbol{D}_{\mathrm{VAL}}[\cdot,\cdot]$; and (3) inference about the credibility of the model as a result of these evaluations. They also require significant attention to the numerical parameter vector $\vec{p}_{\mathrm{N}}$ and its role in determining the numerical accuracy of calculations. Calibration, on the other hand, ultimately is simply an optimization problem in the absence of acknowledged coupling to V&V (we will discuss the influence of this coupling in Section 3). In our language here, calibration depends upon a calibration comparison $\boldsymbol{D}_{\mathrm{CAL}}[\cdot,\cdot]$. The calibration problem can then be summarized as the introduction of an appropriate objective function of the calibration comparison

$$O_{\mathrm{CAL}}(\{\boldsymbol{D}_{\mathrm{CAL}}[M(\vec{p}_{\mathrm{A},i},\vec{p}_{\mathrm{N},i}), B(\vec{p}_{\mathrm{A},i})], i = 1,\ldots,n\}). \tag{6}$$

In other words, $O_{\mathrm{CAL}}(\cdot)$ is a function of the value of a set of calibration comparisons applied to selected benchmarks and appropriately aligned calculations. A weighted aggregation of comparisons as in Eq. (5) could also be used. The task of calibration is then to minimize this objective over a subset of the parameters. While this subset could include the physical alignment parameters $\vec{p}_{\mathrm{A}}$ (see [35]), ideal alignment fixes all of the components of this vector. The assumption of ideal alignment thus means that the calibration minimization can be performed *only* over the numerical parameters $\vec{p}_{\mathrm{N}}$ (recall that examples of components of this vector were discussed in Section 2.3.1). We therefore write the calibration minimization problem as

$$\min_{\{\vec{p}_{\mathrm{N},i}, i=1,\ldots,n\}\subset P_{\mathrm{N}}} O_{\mathrm{CAL}}(\{\boldsymbol{D}_{\mathrm{CAL}}[M(\vec{p}_{\mathrm{A},i},\vec{p}_{\mathrm{N},i}), B(\vec{p}_{\mathrm{A},i})]\}_{i=1,\ldots,n}). \tag{7}$$

To the degree that one believes that components of $\vec{p}_{\mathrm{A}}$ should be included in the calibration problem, this reflects imperfect alignment between the benchmarks and the calculations. It is our experience that in this case the calibration problem is then sometimes perceived as a means of specifying these alignment parameters. We question the logical validity of such a conclusion.

For the shock wave example of Figs. 1–3, under the assumption that $\vec{p}_{\mathrm{A}}$ is well-defined for these experimental benchmarks the substance of our remarks above is that it is unacceptable to consider adjusting the components of $\vec{p}_{\mathrm{A}}$, that is wedge angle, initial gas constitutive parameters, and the incident shock Mach number, to improve agreement of calculations with the benchmarks in Fig. 3. Claiming probabilistic variability in the components of $\vec{p}_{\mathrm{A}}$ does not refute this statement. Known variability, such as in the wedge angle measurement, should instead be used to drive a computational uncertainty quantification that reflects that variability, and which can be used in a more sophisticated validation comparison.

From our point of view, it is permissible to adjust the components of $\vec{p}_{\mathrm{N}}$ to improve agreement with the experimental benchmarks in Fig. 3, although the consequences contribute neither to verification nor validation. (Adjusting $\vec{p}_{\mathrm{N}}$ to better agree with a *verification benchmark* is a logical part of verification, however.) Using the previous example and again assuming that calculations have been performed that are aligned properly with the 18 experimental benchmarks, one form of (7) that could result from such a procedure is

$$\min_{\{\vec{p}_{\mathrm{N},i}, i=1,\ldots,n\}\subset P_{\mathrm{N}}} O_{\mathrm{CAL}}\left(\left|\frac{\Delta p_{\mathrm{r}}}{\Delta p_{\mathrm{inc}}}(\theta_i)^{\mathrm{calc}} - \frac{\Delta p_{\mathrm{r}}}{\Delta p_{\mathrm{inc}}}(\theta_i)^{\mathrm{expt}}\right|, i = 1,\ldots,18\right). \tag{8}$$

In general, if uncertainty is included in the calibration benchmark and the associated calculations, then (7) is an optimization under uncertainty problem. As we have stated previously, for simplicity the reader should view all uncertainty as probabilistically characterized, in which case (7) is a stochastic optimization problem. If only uncertainty in the specification of $\{\vec{p}_{\mathrm{A},i}\}$ and in $B(\vec{p}_{\mathrm{A},i})$ is of concern, this optimization problem is a classical probabilistic parameter estimation problem. We review this problem in Section 3. Given the difficulties of V&V that we have mentioned above, it is likely that code calculation uncertainty will be another contributor to problem (7). Including calculation uncertainty in this calibration problem substantially increases the conceptual difficulty of calibration and we will discuss this further in Section 3.

Generalizing the point we made in the shock wave example, we do not view uncertainty in the specification of the alignment parameters $\{\vec{p}_{\mathrm{A},i}\}$ for a set of benchmarks as a license to use these parameters in calibration. It is more desirable that this uncertainty be quantified and included in the calculated predictions that are of interest. This approach is compatible with the general view that for important computational predictions, quantifying

uncertainty is more important than seeking to reduce it through conservative assumptions and calibration [24,36–39].

2.3.4. Credibility

We now turn our attention to *credibility* of a code for specified applications. Use of the word "credibility" as a description of the quality of a code and its associated calculations can be problematic because of the perception of imprecision associated with the concept. The Oxford English dictionary [40] defines *credible* to mean "capable of being believed." In the particular case of codes, for example, credibility does not *require* truth, accuracy or fidelity. All it requires is belief. However, to be pragmatic and rational, we expect that the credibility of a code for an application rests on good reasons and solid evidence. While evidence is certainly provided by V&V as we have discussed here, reasons for perceived credibility is not necessarily restricted only to V&V. For example, the subjective belief of a scientist using the code will undoubtedly count for something in such a consideration. Nonetheless, our opinion is that objective and precise reasons for credibility are desired over subjective and imprecise reasons, certainly to the degree that belief in code credibility is needed for important decisions. We consider the evidence underlying objective and precise reasons for belief a code is credible for a specified application to be strictly provided by V&V. It is less clear that calibration provides such evidence.

Given the underlying uncertainty in verification, validation and calibration, it is worth restating the impact of this uncertainty in the logic of (The discussion below is a modification of a line of argument on general model uncertainty that has been stressed by John Red-Horse [41].)

- Verification in the comparison of code calculations with verification benchmarks is conditioned on software implementation correctness. By this, we mean that uncertainty in verification, for example in the accuracy of one or more code calculations, depends on the uncertainty in the software implementation. For example, formal techniques such as convergence studies or error estimators may provide evidence that calculations are accurate, but belief in these estimators, or the probability that they are providing accurate information, is dependent upon the probability that there are no software errors corrupting the results. Thus, the numbers that enter into and emerge from a verification comparison can be regarded as probabilistic and conditioned by the probability of software implementation correctness.
  And, to the extent that there is a probabilistic characterization of the likelihood of a bug in the software, this likelihood corrupts definitive acceptance of the results of any verification comparison, even if the comparison is based on mathematically rigorous procedures. This is one reason why techniques that provide further evidence that software is implemented correctly, such as software engineering standards and sensitivity analysis in software testing, formally provide enhanced confidence about the numerical accuracy of the code.
  If the reader does not accept the view that verification has this kind of probabilistic component, the logical dependence of verification upon software implementation still makes complete sense. The deterministic form of our argument is that calculation verification depends upon code verification [6,7]. However, we believe that a correct representation of the issue of verification through benchmarks within a probabilistic characterization of software errors is represented by this logic. The reader should consult Singpurwalla and Wilson [34] for further discussion of the concepts of probabilistic software reliability.
- Similarly, a probabilistic interpretation of validation through the application of validation comparisons is conditioned by the probability that the verification evidence accumulated through verification comparisons is sufficient to provide a high probability that the algorithms and software implementation are correctly and accurately solving the underlying equations of the code.
- Finally, calibration is conditioned by the probability that validation has been successfully performed. We will summarize at least one way to make the logical dependence of calibration upon validation mathematically precise in Section 3.2, when we discuss the implication of acknowledging the so-called model discrepancy in calibration procedures.

The nested probabilistic dependencies that we have suggested above link the three benchmark tasks that we have defined in this paper. Recognition of and incorporation of these dependencies in verification, validation, and calibration seems to be necessary for achieving a quantitative grasp of credibility of code calculations. But while we can argue that this logic is necessary, we are not in a position to claim that it is sufficient for establishing a quantitative picture of code credibility.

A probabilistic interpretation of credibility of the code for stated applications results from our view of conditional probabilistic inference as a product of the application of V&V benchmarks. This credibility is really based upon the results of applying V&V benchmarks rather than calibration benchmarks. We hypothesize the existence of a formalism that quantifies credibility resulting from defined V&V benchmark studies as functions of the comparison results, for example $\boldsymbol{C}^{\mathrm{red}}_{\mathrm{VER}}(\boldsymbol{D}_{\mathrm{VER}}[\cdot,\cdot])$ and $\boldsymbol{C}^{\mathrm{red}}_{\mathrm{VAL}}(\boldsymbol{D}_{\mathrm{VAL}}[\cdot,\cdot])$. The collective information embedded in these quantities, assuming they can be formulated, should summarize the degree to which we believe in the numerical accuracy and physical fidelity of a model for an intended predictive application. This belief will be probabilistic and we hypothesize a Bayesian "degree of belief" interpretation of this probability (see Earman [42], Chapter 2).

It is fair for the reader to question whether such a formalization of "credibility" can ever be achieved. One view of this as a possibility is given in Section 3 below, where we argue that incorporating model uncertainty in calibration in principle can increase our degree of belief in the value of the calibration. We also consider two other examples of credibility formalization.

The first example of a credibility function is a simple Boolean function (Pass/Fail) centered on a verification benchmark. For the Mach flow application presented in Figs. 1–3, verification of the numerical compressible flow algorithms is an important step. One recognized benchmark widely used in the computational shock wave physics community to test compressible flow algorithms is the Sod test problem [43]. This is a one-dimensional Riemann problem that tests the ability of a code to properly compute the elementary wave family (shock, contact discontinuity and rarefaction fan) for a shock tube. This time-dependent problem has a well-known analytic solution as long as the associated waves have not struck the boundaries of the problem.

The verification comparison for this problem is defined as follows:

1. Choose a time $t_S \in (0, t_I]$, where t_I is the first time that a shock, contact, or rarefaction intersect the spatial boundaries of the problem (the spatial domain can be assumed for convenience to be the interval $x \in [0, 1]$.
2. Compute the $l^1[0, 1]$ norm of the difference between the computed density profile at time t_S , $\rho_{\text{calc}}(x, t_S)$, and the discretized benchmark solution $\rho_{\text{Sod}}(x, t_S)$. Thus, we have the comparison:

$$\boldsymbol{D}[M(\vec{p}), \text{Sod}(\vec{p})] = ||\rho_{\text{calc}}(x, t_S) - \rho_{\text{Sod}}(x, t_S)||_{l^1[0,1]}. \quad (9)$$

3. Now define a positive real tolerance ε by some prescription. (This tolerance is typically chosen by subject matter expertise in the absence of formal convergence studies with extrapolation to the limit of zero mesh spacing). Then, we define a credibility function as

$$\begin{aligned} &C^{\text{red}}_{\text{VER}}(D[M(\vec{p}), \text{Sod}(\vec{p})]) \\ &\quad = C^{\text{red}}_{\text{VER}}(||\rho_{\text{calc}}(x, t_S) - \rho_{\text{Sod}}(x, t_S)||_{l^1[0,1]}) \\ &\quad = \begin{cases} \text{Pass} & \text{if } ||\rho_{\text{calc}}(x, t_S) - \rho_{\text{Sod}}(x, t_S)||_{l^1[0,1]} < \varepsilon, \\ \text{Fail} & \text{otherwise.} \end{cases} \end{aligned} \quad (10)$$

The meaning of this statement is: the code is unacceptable if $\boldsymbol{C}^{\text{red}}_{\text{VER}}$ returns the fail value. (The reason for the failure may be algorithmic, code implementation, user error or all three.) It is simply a *necessary* condition to consider using the code that $\boldsymbol{C}^{\text{red}}_{\text{VER}}$ for the Sod problem yields "Pass". The Sod problem is a relatively benign test problem. Any code that is used to compute the experiments with data in Fig. 3 should be expected to pass this test, hopefully in a logically precise sense as suggested above. Passing this test is *not sufficient* to guarantee mathematical and numerical accuracy for the computed results in Fig. 3, however.

In general, credibility functions similar to the example above can be constructed for verification test problems. We have made a leap in formalizing a demand that passing these tests be interpreted as necessary conditions for expectation of mathematical and numerical accuracy of codes for defined applications. In general, we claim that passing such tests is not sufficient for this expectation. How sufficiency can be established through a series of benchmarks with these kinds of credibility functions is an open problem. Establishing sufficiency almost surely requires a standard for code accreditation that is defined by the appropriate subject matter community. We are unaware of the existence of such standards in CS&E. A preferable alternative to defining sufficiency by determination of a panel of experts is to *rigorously prove* that the solution algorithms implemented in the code are solving the corresponding PDEs correctly. We believe that this level of proof has not yet been achieved in general for CS&E.

As a second, highly speculative example of a credibility formalism, we comment that in statistical software reliability theory [34], statistical models of software failures are developed that can be used to predict, for example, the probability that one or more failures will be detected over a future interval of time. The example of the Sod problem used above demonstrates an extension of the notion of a software failure to include failure to pass CS&E benchmark tests. It may then be possible to use probabilistic prediction of future detection of benchmark failures as a means of defining credibility for the code. The execution of such a strategy is far from obvious, but its interpretation in terms of credibility can be made clear. For example, if $F[t_{\text{current}}, t_{\text{current}} + T]$ is the probability of detecting one or more failures in the time interval $[t_{\text{current}}, t_{\text{current}} + T]$ (a period of time T into the future) one might define credibility as: if $F[t_{\text{current}}, t_{\text{current}} + T] > 0.01$, then the code fails; if $F[t_{\text{current}}, t_{\text{current}} + T] \leqslant 0.01$, then the code passes. In words, if the probability of a detected failure is large enough, the code is not credible; if the probability of a detected failure (extended to include benchmark tests) is small enough then the code is credible (say for the purpose of using in validation or for an application). Clearly these kinds of assessments cannot be decoupled from the intended application of the code. The present discussion is only intended to introduce the possibility of using this kind of approach in credibility quantification.

3. Uncertainty and prediction in calibration

3.1. Introduction

In this section, we will further explore the topic of predictive credibility by considering the calibration problem further. Our goal is to expand the scope of the calibration problem from a traditional formulation that

neglects model uncertainty to a formulation that explicitly acknowledges model uncertainty. In the context of the present paper, an examination of the role of model uncertainty in calibration provides additional insight into the discussion of Section 2.

We begin by briefly reviewing classical calibration, and the treatment of benchmark uncertainty in this context. The biggest constraint on the credibility of classical calibration (parameter estimation) to predict response values is the failure to acknowledge uncertainty in the calibrated model (in the language of this literature; for us, a code is an example of a model). Given this understanding, we then review some recent research in Bayesian calibration methods that formally incorporates an analysis of model uncertainty. We call this methodology *Caliberation Under Uncertainty* (CUU).

The focal point of this analysis is the formalization of a *model discrepancy* term that can then be included in Bayesian calibration methods. Model discrepancy is a quantification of the deviation of the model from a chosen set of benchmarks. In other words, model discrepancy is related to the results of V&V as described in Section 2. Our belief is that the study of V&V benchmarks and chosen comparisons yields information that can be used to quantitatively *define or constrain* this discrepancy. It thus seems likely that research on CUU provides, or will provide, useful insight into the meaning and applicability of the type of conditional probability that we suggest in Section 2.3.4, although we will not explore this idea in any detail in this paper.

Finally, we move one logical step further in the process of dealing with model uncertainty in calibration and suggest a possible relationship between CUU and computational learning theory. The advantage that both Bayesian and learning theory methods can bring to model calibration is the formal representation of model discrepancy which can be updated over time as one gains additional information.

We will often use the word "model" in this section, rather than "code," in conformance to the typical language of the topics we review.

3.2. Traditional calibration

3.2.1. General comments about statistical models

The most common example of a calibration method used in practice is linear least-squares regression. While linear regression may not often capture the complex phenomena in CS&E models, we list the assumptions underlying linear regression here as a starting point for understanding the predictive capability one might obtain using linear regression.

Regression models are a class of statistical process models. The underlying assumptions used in statistical process modeling are [44,45]:

1. The underlying process has random variation and is not deterministic.
2. The mean of the random errors are zero.
3. The random errors have constant standard deviation/variance.
4. The random errors follow a normal distribution.
5. The data must be sampled randomly from the underlying process.
6. The explanatory variables (code alignment parameters of Section 2) are observed without error.

One of the most important assumptions in prediction is assumption #4, that the random errors follow a normal distribution. The assumption that the errors have zero mean and constant variance also is important in the formulation of regression models. The mathematical theory for inferences using the normal distribution assumption of error terms is well developed. In practice, the normal distribution often describes the actual distribution of random errors reasonably well. There are a variety of statistical tests to check for normality of errors. If this assumption is violated, then the inferences made about the process based on this assumption may be incorrect.

3.2.2. Linear least-squares regression

The most widely used method to estimate parameters in a model is to use a linear least-squares regression. In a regression model with one dependent variable y (that we assume for simplicity is a single scalar) and multiple independent variables $x_j,\ j = 1,\ldots,k$, the linear model is formulated as

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k + \varepsilon, \tag{11}$$

where ε represents random error associated with the model (some textbooks emphasize that this error is observational error; others state that all unexplained variation in y caused by important but unincluded variables or by unexplainable random phenomena is included in the random error term). Usually, the random error is assumed to be normally distributed with mean 0 and variance σ^2, written $\varepsilon \sim N(0, \sigma^2)$.

Least-squares regression minimizes the sum of squares of the deviations of the y-values about their predicted values for all the data points. Thus, for n data points, the *Sum of Squares of the Errors* (*SSE*) is

$$\mathrm{SSE} = \sum_{i=1}^{n} [y_i - (\beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k)]^2. \tag{12}$$

The quantities $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_k$ that minimize the SSE are called the *least-squares estimates* of the parameters $\beta_0, \beta_1, \ldots, \beta_k$ and the resulting prediction equation is

$$y = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \cdots + \hat{\beta}_k x_k + \varepsilon, \tag{13}$$

where the "hat" notation can be read as "statistical estimator of." Thus, $\hat{y}$ is the least-squares estimator of the mean of y, $E(y)$, and $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_k$ are estimators of the parameters $\beta_0, \beta_1, \ldots, \beta_k$.

In the notation of Section 2, a given set of data points y_i are benchmark data, and the regression equation (11) is the

numerical model, or code. Thus, linear regression is a calibration formulation where we demand $B(\vec{p}_i) = y_i = M(\vec{p}_i) + \varepsilon_i$. The alignment parameters ("explanatory variables") are $\vec{p}_{A,i} = (x_1^i, \ldots, x_k^i)$, while the remaining parameters are $\vec{p}_N = (\beta_0, \beta_1, \ldots, \beta_k)$. (What could be a very complex family of "numerical" parameters for a code are simply the coefficients in the linear model in this case.) Finally, calibration is accomplished by minimizing the SSE, which means if:

$$\boldsymbol{D}_{\mathrm{CAL}}[M(\vec{p}_i), B(\vec{p}_i)] \equiv [y_i - \beta_0 + \beta_1 x_1^i + \cdots + \beta_k x_k^i]^2,$$

$$O_{\mathrm{CAL}}(\{\boldsymbol{D}_{\mathrm{CAL}}[M(\vec{p}_i), B(\vec{p}_i)]\}_{i=1,\ldots n}) \equiv \sum_{i=1}^{n} \boldsymbol{D}_{\mathrm{CAL}}[M(\vec{p}_i), B(\vec{p}_i)],$$

then

$$\min_{\beta_0,\beta_1,\ldots,\beta_k} \mathrm{SSE} = \min_{\beta_0,\beta_1,\ldots,\beta_k} O_{\mathrm{CAL}}(\{D_{\mathrm{CAL}}[M(\vec{p}_i), B(\vec{p}_i)]\}_{i=1,\ldots,n}), \tag{14}$$

which is in the form of Eq. (7). The minimum occurs at $(\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_k)$.

Regression models calibrated in this fashion are used for prediction in a variety of ways without explicit characterization of model uncertainty. Some of the most common prediction tasks are to:

1. Predict a *response* at a particular set of *input variables* different than the benchmark data. The input variables are also called *predictor*, *independent*, or *explanatory* variables. This is done by substituting a particular value of $\vec{x}$, say $\vec{x}'$, into the regression formula with the fitted coefficients: $\hat{y}' = \hat{\beta}_0 + \hat{\beta}_1 x_1' + \cdots + \hat{\beta}_k x_k'$. This is interpreted as: the mean or expected response at input x' is given by $\hat{y}'$.
2. Describe the *confidence interval* (CI) for the mean response at a particular set of input values (given in step 1). The CI range (e.g., a 95% CI) is the range in which the estimated mean response for a set of predictor values is expected to fall. This interval is defined by lower and upper limits, calculated from the confidence level and the standard error of the fits. The CI is around the $\hat{y}'$, and is interpreted as: the CI will contain the true mean at input $\vec{x}'$ a certain percentage of time (e.g., 95% for a 95% CI).
3. Describe the *prediction interval* (PI) at a particular set of input values. The PI is the range in which the predicted response for a new observation is expected to fall in a statistical sense. The PI differs from the CI in that the PI is an interval in which a particular response for input $\vec{x}'$ is expected to fall, and the CI is the interval in which the mean response for input $\vec{x}'$ is expected to fall. The PI is defined by lower and upper limits, which are calculated from the confidence level and the standard error of the prediction. The PI is always wider than the CI because of the added uncertainty involved in predicting a single response versus the mean response.

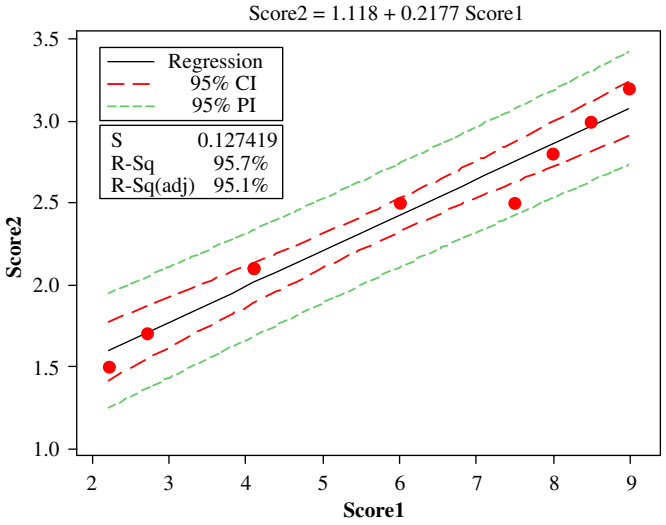

Fig. 4. Confidence versus prediction intervals in linear regression.

The formulas used to calculate the CI and the PI are found in most statistical texts [45].

An example of CI vs. PI for a linear regression model is shown in Fig. 4. There, the discrete dots are (calibration) benchmarks, the black solid line is the linear regression resulting from the calibration problem (Eq. (14)), the dashed red lines are 95% CIs, and the dashed green lines are 95% PIs. Note that these intervals are around individual input points. Methods such as Scheffé or Bonferroni confidence intervals give CI estimates around the regression coefficients [45].

Linear regression is used extensively in practice because many processes are well-described by linear models, or at least well-approximated by a linear model over a constrained domain of $\vec{x} = \vec{p}_A$. The statistical theory associated with linear regression is well-understood and allows for construction of different types of easily interpretable statistical intervals for predictions, calibrations, and optimizations. These statistical intervals can then be used to give clear answers to scientific and engineering questions.

The main disadvantages of linear least squares are limitations in the shapes that linear models can assume over broader ranges of $\vec{p}_A$, and sensitivity to outliers. Additionally, the method of least squares is very sensitive to the presence of unusual benchmark data points in the collection used to calibrate a model. One or two outliers can sometimes seriously skew the results of a least-squares analysis.

3.2.3. Nonlinear least-squares regression

Nonlinear least-squares regression extends linear least squares regression for use with a much larger and more general class of functions. Almost any function that can be written in closed form can be incorporated in a nonlinear regression model [46]. Unlike linear regression, there are

very few limitations on the way parameters can be used in the functional part of a nonlinear regression model. The way in which the unknown parameters in the function are estimated, however, is conceptually the same as it is in linear least-squares regression.

In nonlinear regression, the nonlinear model between the response y and the predictor $\vec{x}$ is given as: $y = f(\vec{x}, \vec{p}_N) + \varepsilon$, where ε is the random error term and $\vec{p}_N$ is the auxiliary parameter vector as in Section 2. For example, one could have a benchmark data set, $\{(\vec{x}_i, y_i), i = 1, \ldots, n\}$ where

$$\vec{x} = x, \quad p_N = (p_1, p_2),$$
$$y_i = f(x_i, p_N) = p_1[1 - e^{p_2 x_i}] + \varepsilon_i.$$

The goal of *nonlinear regression* is then to find the optimal values of $\vec{p}_N$ to minimize the function $\sum_{i=1}^{n}(y_i - f(\vec{x}_i, p_N))^2$. This problem can be cast in the same form as Eq. (14) and so is also compatible with our discussion in Section 2.

The value of $\vec{p}_N$ that minimizes the sum of squares is $\hat{p}_N$, and has an estimated covariance matrix given by $\mathrm{Cov}(\hat{p}) = s^2(W'W)^{-1}$, where W is an $n \times k$ (in general) matrix of first derivatives of the assumed nonlinear model $f(\vec{x}, \vec{p}_N)$ evaluated at $\hat{p}_N$, and $s^2 = \mathrm{SSE}/(n-k) =$ estimator of σ^2. In terms of prediction, equations similar to linear regression are used to calibrate the nonlinear model with the benchmark data. If one is predicting the mean response at a particular value of $\vec{x}_0$ not in the benchmark data, CI and PI analogous to those mentioned for linear regression can be developed.

Nonlinear regression is formally similar to linear regression in the sense of calibration presented in Section 2 in that the resulting abstract formulation is still $B(\vec{p}_i) = M(\vec{p}_i) + \varepsilon_i$, with $\vec{p} = (\vec{x}, \vec{p}_N)$ and $\vec{p}_N$ determined through a calibration optimization problem. The main differences are that $M(\vec{p}_i)$ is now a nonlinear function (it could be as complex as a code, of course), and the auxiliary parameters $\vec{p}_N$ no longer enter the formulation as coefficients of a linear equation, as in the case of linear regression. In nonlinear regression, the coefficients are usually more closely tied to a physical quantity than in linear regression, as a matter of conventional practice. Nonlinear regression requires an optimization algorithm to find the value of $\vec{p}_N$ that minimizes the sum of squares analogous to Eq. (14), which rapidly increases the technical difficulty of the calibration procedure. Specialized nonlinear least-squares optimization algorithms have been designed to exploit the special structure of a sum of the squares objective function and solve these complicated optimization problems. Nonparametric regression techniques, such as moving least squares, are also used as statistical models for data. In some cases, nonparametric methods may provide better local "fits" of the data than nonlinear regression methods. However, nonparametric regression methods are more localized methods (similar to radial basis functions), so we do not see them having the same role in calibration as a global regression fit.

3.3. Calibration under uncertainty

The regression methods outlined above are used in calibrating parameters of models, taking into account uncertainty in the calibration benchmark data but assuming no uncertainty in the model itself. However, model uncertainty also influences the credibility of predictions based on the calibration. The Bayesian statistics community has developed formal statistical methods that address model uncertainty. This is called CUU. One important approach is that of Kennedy and O'Hagan [47], hereafter referred to as KOH. They formulate a model for calibration data that includes an experimental error term (similar to the random error term in linear regression) and a model discrepancy term, which quantifies the deviation of the model from a chosen set of benchmarks. Model discrepancy is represented by a Gaussian process (GP) (think of it as a random field; this representation is more sophisticated than a constant bias term). KOH then use a Bayesian approach to update the statistical parameters associated with the discrepancy term and with the model parameters. The purpose of updating is to reduce uncertainty in the parameters through the application of additional information. Reduced uncertainty is presumed to increase the predictive content of the calibration. Examples of application of the KOH approach are given in [48,49].

Below, we give a very brief outline of the key points in the KOH approach. The interested reader should peruse the references.

3.3.1. Kennedy and O'Hagan formulation

The KOH approach uses the idea of GP. GP models are used in response surface modeling, especially response surfaces which "emulate" computer codes. GP have also been used in conjunction with Bayesian analysis for regression problems and for calibration, and for estimation and prediction in geostatistics and similar spatial statistics applications [50].

A GP is defined as follows [51]: A stochastic process is a collection of random variables $\{Y(\vec{x})\vec{x} \in X\}$ indexed by a set X (in most cases, X is $\mathbb{R}^d$, where d is the number of model input parameters). A GP is a stochastic process for which any finite set of Y-variables has a joint multivariate Gaussian distribution. Note that the random variables Y are indexed by $\vec{x}$: a set such as $\{Y(\vec{x}_1), \ldots, Y(\vec{x}_k)\}$ has a joint Gaussian probability distribution. A GP is fully specified by its mean function $\mu(\vec{x})$ and its covariance function $C(\vec{x}, \vec{x}')$. The mean function may be a polynomial, a sum of weighted basis functions, etc. The covariance function is often defined as an exponential function of the form: $C(\vec{x}, \vec{x}') = v_0 \exp\{-\sum_{u=1}^{d} \rho_u^2(\vec{x}_u - \vec{x}'_u)^2\}$, to reflect the fact that nearby input points have highly correlated outputs. Prediction using GP models involves matrix algebra on the inverse of the covariance matrix of all the data points.

KOH's approach also relies on the idea of Bayesian updating. The basic concept in Bayesian analysis is to

combine "prior" information (in terms of a distribution on a parameter, where the distribution itself is characterized by "hyperparameters") and actual data (through a likelihood function) to obtain a "posterior" estimate of various parameters. Combining GP and a Bayesian approach, one places a prior probability distribution over possible mean and covariance functions of the GP and generates a posterior distribution on the GP parameters based on the data. One can then sample from the posterior distribution to obtain various estimates of the GP prediction. We do not address the issue of Bayesian analysis/Bayesian updating in this paper due to space limitations. Gelman et al. [52], and Press [53] provide primers on Bayesian topics. One way to generate a posterior distribution is through Markov Chain Monte Carlo sampling (MCMC). Specific references for MCMC sampling include [54,55].

With this very brief background in GP models and Bayesian analysis, we proceed to the KOH model for CUU. We consider a computer model (a computer code in the language of Section 2) denoted $\eta(\vec{x},\vec{t})$, where $\vec{x} = (x_1,\ldots,x_q)$ plays the role of the previously discussed alignment parameter vector $\vec{p}_{\mathrm{A}}$ and the vector $\vec{t} = (t_1,\ldots,t_r)$, the calibration parameters in the language of KOH, plays the role of the auxiliary parameters $\vec{p}_{\mathrm{N}}$. KOH assume that these calibration parameters take fixed but unknown "true" values $\vec{\theta} = (\theta_1,\ldots,\theta_r)$ differentiated from the fixed values $\vec{t}$ which are set as inputs when evaluating the model. The model $\eta(\vec{x},\vec{t})$ has uncertainty, with the assumed "true" value of the real physical process simulated by the model for the input $\vec{x}$ denoted by $\zeta(\vec{x})$. The outputs from N runs of the computer model are then given by $\{y_i = \eta(\vec{x}_i,\vec{t}_i), i = 1,\ldots,N\}$. Observed benchmark data for the physical process of interest may consist of $n<N$ data points, which we denote as $\vec{z} = (z_1,\ldots,z_n)$ (we assume for simplicity that the benchmark data are scalars). In KOH's formulation, they represent the relationship between the observations, the true process, and the computer model output by the fundamental equation:

$$z_i = \zeta(\vec{x}_i) + e_i = \rho\eta(\vec{x}_i,\vec{t}_i) + \delta(\vec{x}_i) + e_i, \tag{15}$$

where e_i is the observation error (uncertainty in the benchmark data) for the ith observation, ρ is introduced as an unknown regression parameter, and $\delta(\vec{x})$ is the *model discrepancy* or model inadequacy function and is treated as *independent* of the code output $\eta(\vec{x},\vec{t})$. The model discrepancy term is an empirical description of model inadequacies and may include factors such as code bugs, poor numerical accuracy due to insufficient grid resolution, poor alignment with respect to a benchmark, incomplete physics, and so on. Of particular concern to us in this paper, poor comparison with benchmarks is a direct measure of an underlying model (in our case, code) discrepancy. The KOH model discrepancy is a statistical construct that has been primarily applied for relatively simple computational applications compared to our CS&E concerns.

A few comments about this approach are in order. The KOH formalism is a highly parameterized method for incorporating model uncertainty in classical statistical calibration procedures, with both the code output $\eta(\vec{x},\vec{t})$ and $\delta(\vec{x})$ represented as GP to allow inference. The error term e_i should include both residual variability as well as observation error, but KOH do not strictly address residual variability in the benchmark data currently. They assume that e_i is normally distributed as $N(0,\lambda)$ (this assumption rules out systematic errors). Assumption of a constant value of the introduced regression parameter ρ implies that the underlying benchmark process $\zeta(\vec{x})$ is stationary, which is unrealistic in many applications.

Restating several hints made above, the KOH model given in (15) can be mapped to the formalism defined in Section 2 as follows: The code parameter vector decomposition has $\vec{p}_{\mathrm{A}}$ corresponding to $\vec{x}$, and $\vec{p}_{\mathrm{N}}$ corresponding to $\vec{\theta}$, with specific values alternatively labeled by $\vec{t}$. The benchmarks are given by $\vec{z}$, that is $B(\vec{p}_i) = z_i$. The code output $M(\vec{p}) \equiv M(\vec{p}_{\mathrm{A}},\vec{p}_{\mathrm{N}})$ is denoted by the KOH "model" $\eta(\vec{x},\vec{t})$. The distinction between the KOH method and the traditional regression approach summarized above is the addition of the model discrepancy term in (15). Thus, in terms of the discussion of Section 2, in the KOH approach, we have the benchmark comparison process formalized as: $B(\vec{p}_i) = M(\vec{p}_i) + \Delta M(\vec{p}_i) + e_i$. The model discrepancy term $\Delta M(\vec{p}_i)$ in the KOH approach is a function only of inputs (alignment parameters) $\vec{x}$. In model V&V, an important goal is then to quantify the model discrepancy term. In the KOH approach, the next step taken is to use the benchmark data to update assumed prior distributions on parameters in the model and model discrepancy terms with Bayesian methods. This type of calibration is not an optimization in the sense of minimizing the SSE (as in Eq. (14)) or an equivalent metric. Rather, the KOH calibration is an updated picture (a posterior distribution) of the parameter values governing the model term and model discrepancy term based on Bayesian principles which is aimed at reducing uncertainty in these parameters. This is understood to improve the quality of predictions made with the model, not through some kind of tuning of model parameters through minimization of a measure of discrepancy with the benchmark data, but through more refined quantification of uncertainty in the model parameters. Clearly, some sense of when it is important to perform the latter rather than the former process on the parameters should serve to drive this methodology.

The methods for attacking the generation of posterior distributions for the parameters in this formalism are complicated. We direct the interested reader to the original work of KOH, as well as a detailed summary in Swiler and Trucano [56].

3.3.2. Comments on implementation issues

We are presently investigating the feasibility of using a GP formulation such as provided by KOH as a practical

calibration method for engineering design problems. Campbell [57] has also reviewed KOH's work with an emphasis on implementation. She concluded that information about model quality gained through the formulation of a model discrepancy term could be useful, but that a user should be careful in situations of limited observational data and "avoid exaggerating the contribution of the Bayesian updating process." [57, p. 2].

As an exercise, we developed a GP emulator for the Rosenbrock function [56], a test function used in the optimization community. We observed that as we added data points, the covariance matrix $C(\vec{x}, \vec{x}')$ underlying the GP characterization became extremely ill-conditioned, and the inverse covariance matrix needed for prediction with posterior updates could not be generated (the ratio of largest and smallest eigenvalues in the covariance matrix was 10^{16}!) Note that this behavior intuitively seems wrong: more data should always be better in terms of creating a response model or performing prediction. But in GP models, if points are close together in the input space, the resulting covariance matrix can have rows that are nearly linearly dependent, and the inversion falls apart. There are methods to address this (e.g., singular value decomposition), but our experience stands as a caution. Ultimately, we believe this difficulty is algorithmic, rather than conceptual.

We offer one further caution about these methods. Thus far, we have found that it is difficult to separate the model discrepancy and the observational error term, e_i, unless one has very good information about measurement error in the benchmark data. Also, the KOH framework requires that the user have reasonably good prior estimates for the parameters of both the model emulator GP and the model discrepancy GP. In practice, this is not always the case. Finally, there is a software implementation issue. Most public domain software (e.g., Netlab, Flexible Bayes Modeling (FBM) software), allows one to create a GP and update the hyperparameters governing the GP, usually with a MCMC method. However, the ability to perform simultaneous updating on two GP models coupled by Eq. (15) to obtain the parameters governing both the η and δ terms is not generally available. Finally, MCMC methods which are commonly used for these types of problems require the user to have a fair amount of statistical knowledge about the form of the posterior (in terms of a "proposal distribution" used to generate the Markov chain), certainly in order to make evaluation more efficient. MCMC methods require a lot of tuning parameters, such as step sizes and leaping parameters, and it is not trivial to tune the MCMC to obtain a recommended acceptance rate of 25%, for example. And testing convergence of MCMC methods is difficult. There are some convergence diagnostics available [54] but they test if the chain has "settled" out and do not really test if the Markov chain has converged to the "true" underlying posterior distribution. We have tested cases where two different chains produced substantially different posterior distributions. It appears to us that tuning MCMC performance and improving convergence diagnostics is a research question of interest.

3.3.3. Conclusions

Our interest in the work of KOH is dominated by the potential for enhanced prediction resulting from CUU methods and on the connection of these methods to V&V through the model discrepancy. Overall, our conclusions to date from this work in progress are the following: GP models are powerful surrogates or response surface "emulators" for computationally expensive codes. Implementing them requires some knowledge about the data set used and what data will have to be discarded to make the covariance matrix well-conditioned; or additional formulations are needed that are robust to poor covariance conditioning. KOH's formulation of "observation = model + discrepancy + error" is very important because it explicitly separates the model discrepancy term from the model itself. The GP assumption of the model discrepancy needs further examination (and this might result from focusing V&V results within a CUU framework), but in general, GP models are extremely flexible at representing a wide variety of functional relationships. The additional assumption that these GP models are governed by parameters that can be updated using Bayesian methods adds a great deal of computational complexity to the picture. The formulation of the joint posterior is difficult. Even if one does not try for analytic solutions but uses MCMC methods, there are many issues to resolve around numerical performance, such as convergence of the MCMC to the correct underlying posterior and determination of tuning parameters, etc.

At this point, we see some interesting paths for further investigation. One is using KOH's formulation expressed in (15) but calculating the parameters by Maximum Likelihood Estimation (MLE) methods instead of Bayesian updating. Dennis Cox has pursued this approach [58] and it removes the difficulties associated with posterior generation (such as via MCMC). Another is to look at the first term in Eq. (15), the model emulation term, and replace it with another type of surrogate, perhaps a lower fidelity or reduced order model. This has the advantage of "simplifying" the estimation in that one is solely focused on calculating the parameters for the GP model, but it may introduce limitations in terms of the capability to predict.

A final thought about the KOH method with respect to calibration is worth emphasizing. The updating of parameters governing a GP model of the code and/or a GP model of the model error does provide new, "calibrated" estimates of the hyperparameters of the parameter distributions. However, it does not directly "optimize" these parameters in the sense that regression does: it merely characterizes them. In many cases, the data may not change the parameters significantly in the Bayesian updating process. The method based on learning theory in the following section also characterizes and tries to provide a good estimate of the model discrepancy term but

does not directly "optimize" parameters; rather it is an adaptive method of understanding the discrepancy term.

3.4. Extension of the context of calibration—learning theory

3.4.1. Introduction

We have suggested that the complexity of calibration increases (significantly!) as we incorporate formal measures of model uncertainty into the procedures. In this section we would like to further abstract this issue from a somewhat different perspective, that of the mathematics of generalization, also called *supervised learning* [59].

We have observed that the calibration problem, defined in Section 2 as minimization of a complex functional such as $\min_{\{\vec{p}_{\mathrm{N},i},i=1,\ldots,n\}\subset P_{\mathrm{N}}} O_{\mathrm{CAL}}(\{\boldsymbol{D}_{\mathrm{CAL}}[M(\vec{p}_{\mathrm{A},i},\vec{p}_{\mathrm{N},i}), B(\vec{p}_{\mathrm{A},i})], i = 1,\ldots,n\})$ should include a formalization of model uncertainty, for example as in the model discrepancy term, which we here write as $\Delta M(\vec{p})$, discussed in Section 3.3. We have also claimed without detailed analysis that the evaluative nature of V&V manifested in the information gathered from benchmark comparisons $\boldsymbol{D}_{\mathrm{VER}}[\cdot,\cdot]$ and $\boldsymbol{D}_{\mathrm{VAL}}[\cdot,\cdot]$ contributes quantitative information about the code (model in KOH terms) discrepancy $\Delta M(\vec{p})$.

One way to conceptually integrate these procedures can be summarized as follows. As we stated earlier, the code $M(\vec{p})$ defines a mapping (function) of the input parameter space to the output space:

$$M(\vec{p}) : \boldsymbol{P} \equiv \boldsymbol{P}_{\mathrm{A}} \times \boldsymbol{P}_{\mathrm{N}} \rightarrow \boldsymbol{O}.$$

The code discrepancy is also such a mapping:

$$\Delta M(\vec{p}) : \boldsymbol{P} \equiv \boldsymbol{P}_{\mathrm{A}} \times \boldsymbol{P}_{\mathrm{N}} \rightarrow \boldsymbol{O}.$$

(Note that we no longer assume the restriction of KOH that $\Delta M(\vec{p})$ only depends on $\vec{p}_{\mathrm{A}}$.) For a code that is credible for a specific application associated with a specific choice of $\vec{p}$ we want $\Delta M(\vec{p})$ to be small. The decision to calibrate is simply to force this to happen on a set of benchmarks (which we now write in a form to be compatible with learning theory notation) $\boldsymbol{T} \equiv \{\vec{p}_{\mathrm{A},i}, B(\vec{p}_{\mathrm{A},i}), i = 1,\ldots,n\}$ through adjustment of appropriate parts of $\vec{p}$. V&V should guide this process through a quantification of $\Delta M(\vec{p})$ specifically on the set $\boldsymbol{T}$ and through correlation of the features detected in V&V with details of the model structure, including algorithm design, software implementation and physics conceptual models. This evaluation should include quantification of the uncertainty in $\boldsymbol{T}$ and $M(\vec{p})$.

From this perspective, there is then some hope that all of this knowledge, coupled with statistical calibration procedures, will provide a prediction of $\Delta M(\vec{p})$ off of the benchmark set $\boldsymbol{T}$ that we have some quantitative reason to believe is credibly small. In the language of supervised learning, $\Delta M(\vec{p})$ is a *target relationship* between $\boldsymbol{P}$ and $\boldsymbol{O}$, and $\boldsymbol{T}$ is a *training* set. Via benchmark tasks and the associated inference, we *learn* about $\Delta M(\vec{p})$ from its restriction to the training set, and use learning methods to extend this relationship beyond the training set. In the ideal, from a perfectly constructed training set $\boldsymbol{T}$ and the right learning methods, we can extend $\Delta M(\vec{p})$ to the application domain. If $\Delta M(\vec{p})$ is then sufficiently small on the application domain within this context, we have a formalization of the credibility of applying $M(\vec{p})$ for prediction, i.e. to values of $\vec{p}$ different from the training set. This formalization will be probabilistic since the learning framework must be probabilistic (although this could be subjective, that is not using a frequency-based probability interpretation). We emphasize that calibration is not really fundamental to the learning theory generalization of $\Delta M(\vec{p})$ from a benchmark training set. However, we are allowing for the possibility that $\Delta M(\vec{p})$ may not be small enough for the intended application on the training set and that an attempt (calibration) will be made to reduce it. Note that neither the KOH Bayesian method nor the learning theory approach calibrates parameters directly in terms of minimizing the SSE or similar objective functions of the parameters. Rather, these methods improve the characterization of the model discrepancy term $\Delta M(\vec{p})$. In the KOH approach, prior distributions on parameters are assumed, and then calibration is an updated picture (a posterior distribution) of these parameter values governing the model term and model discrepancy term based on Bayesian principles. In learning theory, something about $\Delta M(\vec{p})$ is learned from the training set, and this relationship is then extended to an application domain.

We now make some comments about this view of V&V and calibration:

1. Because of the role of uncertainty in all aspects of V&V, the representation of these problems in a learning theory context stated above is rather natural from a philosophical perspective.
2. Learning is a readily recognizable feature of the human interaction with complex CS&E codes, as any experienced practitioner will recognize. In particular, this learning is expressed in the understanding of variation and choice of the code parameters most likely to be used in formal calibrations. Learning is more than calibration in the sense that experience provides subjective or otherwise projection of the likely code discrepancy that arises when the code is applied away from the experience base of benchmark (or training) data. This empirical experience seems to us to be worth understanding more formally in the context of V&V, as well as in CUU.
3. One of the recognized mathematical problems of learning theory [60,61] is the nature of the behavior of the choice of generalization of $\Delta M(\vec{p})$ (restricting to our context), also called the *hypothesis*, as the training set increases. Any understanding of this depends on the nature of the training set as well as technical restrictions on the space that $\Delta M(\vec{p})$ may be chosen from. When benchmarks in the training set are validation benchmarks, this problem expresses concisely one of the most important problems in experimental validation—what is the value of performing new validation experiments?

These authors also note that for a fixed training set there is an optimal complexity of the space that $\Delta M(\vec{p})$ may be chosen from. This hints at the possibility that mathematical learning theory can provide some guidance as to how to choose a $\Delta M(\vec{p})$ that is most usefully generalizable beyond the specified training (benchmark) data.

4. One more time we emphasize our original decomposition of the parameter vector that the computational model depends upon. The goal is generalization of $\Delta M(\vec{p})$, that is interpolation or extrapolation to other values of $\vec{p}_{\mathrm{A}}$, not simply reducing the size of $\Delta M(\vec{p})$ on the specified training set $\boldsymbol{T}$ (the set of benchmarks). In this way we make visible the intuitive fact that *generalization* is important for credibility, not simply point accuracy restricted to the training set. Calibration in isolation from V&V is satisfied with the narrow goal of only making $\Delta M(\vec{p})$ small. Developing methods for understanding generalization independent of calibration is important, and this is one of the reasons to study CUU.

In principle, a learning theory perspective on V&V and calibration can be readily adapted to direct application of supervised learning to the code $M(\vec{p})$. In this case, $\Delta M(\vec{p})$ then acts in some sense as intermediate information that helps to guide the generalization of $M(\vec{p})$ by means of calibration to reduce $\Delta M(\vec{p})$ on the training set. Our belief is that for something as complex as a code, it is better to restrict attention in the learning theory context to the discrepancy $\Delta M(\vec{p})$. Certainly, we should separate the problem of learning about $\Delta M(\vec{p})$ from the problem of trying to minimize it as a mechanism for confidently generalizing $M(\vec{p})$. For one thing, the more likely it is that $M(\vec{p})$ is credible, in other words, that it is generalizable off the training set, then the more likely it is that $\Delta M(\vec{p})$ will have a simple structure, for example to be relatively smooth and flat as a function of $\vec{p}_{\mathrm{A}}$. In this case, it is believable that training will provide a useful generalization of $\Delta M(\vec{p})$. Obviously, this argument requires significant elaboration; we are currently engaged in research to explore these issues in greater detail. Some further elements of a learning theory perspective are briefly summarized below (see also [62]).

3.4.2. Vector of learning parameters

While there are many types of learning, the focus herein is on *parameter learning*. By this we mean that there is a vector of parameters $\vec{p}_{\mathrm{L}}$ in the code with components that would include a calibration vector (e.g., the vector θ used in the KOH approach) as well as those components of the alignment parameter vector $\vec{p}_{\mathrm{A}}$ that are not precisely known. To be compatible with the notation of this paper, we can assume that the components of $\vec{p}_{\mathrm{L}}$ lie in the vector $\vec{p}_{\mathrm{N}}$ introduced in Section 2, thus implying that ill-defined components of $\vec{p}_{\mathrm{A}}$ (for whatever reason!) are removed from the alignment process and entered as components in the vector that could be calibrated. We thus assume that the components of $\vec{p}_{\mathrm{L}}$ cannot therefore be assigned in a straightforward manner by alignment or user judgment. This parameter vector would be learned as part of a benchmark process through the application of machine learning techniques. Some examples of the type of parameters that are found in $\vec{p}_{\mathrm{L}}$ are:

- Unobservable parameters, such as unmeasured quantities in a validation experiment, that appear in the code.
- Component-to-component interaction parameters, such as effective parameters in a constitutive model implemented in the code.
- Model selection parameters, for example "multipliers" used in a code to adjust the relative influence of particular model components in a code.
- Numerical parameters $\vec{p}_{\mathrm{N}}$.

3.4.3. Types of computational learning

Learning theory is broad and tends to be highly application specific. An attempt is made here to provide a brief overview of the main goals of learning theory in the context of parameter learning so as to contrast the above discussion that focused on statistical calibration. We choose to guide this discussion through the description of problems of particular interest in computational learning.

3.4.3.1. Statistical interpolation. The abstract goal here is to determine a mathematical manifold (essentially a set of high-dimensional surfaces), expressed as a function of $\vec{p}_{\mathrm{L}}$, using statistical tools. The KOH CUU method is an example of a method which addresses this learning goal; many other methods can be found in texts such as that by Hastie et al. [63]. These methods work by optimizing the interpolation result with respect to a statistical measure of fit, such as likelihood. Bayesian extensions are also possible, yielding probability densities for the parameter vector. The idea of statistical interpolation is to make best use of the information contained in the training data set to guide optimal selection of $\vec{p}_{\mathrm{L}}$ for given choices of $\vec{p}_{\mathrm{A}}$ that differ from the training set and can be viewed as interpolating or extrapolating the training set. Subject matter expertise (domain expertise in the language of learning theory) is needed to define the parameter bounds and to assist in statistical reduction of the parameter space, but the interaction between the domain expert and the statistical tools is relatively weak in this problem. In other words, the emphasis is likely on the statistical properties of the training data set, not domain knowledge.

3.4.3.2. Elicitation of domain knowledge. The learning goal here is to expand the information beyond the training data set by using domain experts (see [64–66]). The learning constraint, as mentioned earlier, is that the results of this interaction should be restricted to the parameter vector $\vec{p}_{\mathrm{L}}$ and should not, for instance, require modifications to complex code. Hence, the type of domain experts needed

here are not necessarily those who write the code, but those who are experts in using the code. Except for problems involving simple or well-understood systems, it can be assumed that such experts will be available.

3.4.3.3. Integrated learning. Statistical interpolation and domain knowledge can be combined in an integrated approach: this is called *integrated learning*. The goal is to be able to elicit knowledge from domain experts and use this knowledge in a statistically consistent manner with the training data. What is critical here is a close interaction between the domain expert and the training data set to allow for insightful interpretation and generalization by the expert in the context of the data. This immediately precludes the use of legacy methods such as case-based reasoning and knowledge-based systems, because they are developed for non-interactive environments where the experts deposit some form of knowledge into a computational framework and this knowledge is subsequently applied to a modeling problem. This also requires substantial extensions and generalizations of the domain expertise interaction used in statistical learning, as described above. Further details of this approach have been summarized in Igusa and Trucano. Other integrated methods, including those using agents, are currently being explored.

3.4.4. Parameter reduction and metrics

3.4.4.1. High-dimensional parameter problems. Statistical interpolation can operate directly in high-dimensional spaces if the training data set is sufficiently large. Since the size of such sufficiently large training sets grows exponentially with parameter dimension, practical limitations may require some reduced form of statistical interpolation. There is a large collection of statistical reduction techniques that are available, nearly all of which require interaction with domain expertise. For instance, the simplest reduction techniques (such as ANOVA [67]) eliminate parameters from the interpolation scheme through sensitivity analysis, making the interpolation invariant with respect to these parameters. Domain expertise is needed to indicate whether the system is truly insensitive with respect the eliminated parameters.

Integrated learning couples domain expertise more closely to the parameter reduction process. The essential idea here is to replace a high-dimensional, low-level parameter vector with a relatively low-dimensional, high-level feature vector. The components of the feature vector are simple functions of the original parameter vector that are defined through the interaction of the domain expert with the computational learning method (through the application of projection pursuit regression, for example; see [63,68]). This is in contrast to statistical reduction where a low-dimensional vector is obtained in terms of the original parameter vector using generic functions (e.g., sample averages and variances), subsets of components, or linear combinations (e.g., principal components).

3.4.4.2. Complex observational metrics. In some simple systems, benchmark data sets $\{y_{ij} : j = 1, \ldots, m\}$, say for experiments $i = 1, \ldots, n$, each containing m data values may be characterized by small m and a uniformity of data types (including experimental repeats). In such cases, the benchmark comparisons $\boldsymbol{D}_{\mathrm{VER}}[\cdot,\cdot]$, $\boldsymbol{D}_{\mathrm{VAL}}[\cdot,\cdot]$ and $\boldsymbol{D}_{\mathrm{CAL}}[\cdot,\cdot]$ could be defined by the null statistical form, namely, weighted least squares similar to the discussion in Section 3.2:

$$\boldsymbol{D}[\{y_{ij}\},\{y_{ij}^{(M)}\}] = \sum_{i=1}^{m} w_j[y_{ij} - y_{ij}^{(M)}]^2, \quad j = 1, \ldots, m, \tag{16}$$

where $y_{ij} = y(\vec{p}_{\mathrm{A},ij})$, $y_{ij}^{M} = M(\vec{p}_{\mathrm{A},ij})$. As the data sets increase in size (e.g., time-dependent data), then the above function can become dominated by irreducible noise. (An example is where the data y is given by the displacement response of a simple linear oscillator at discrete time intervals along with the forcing function, and the measurements of the forcing function are corrupted by a small amount of white noise. If the model M is of a linear oscillator subjected to the measured forcing function, then the null statistical metric would provide poor resolution, particularly for large time durations.) With limited interaction with a domain expert, it is possible to improve the comparison, using a reduced form of the data. For time-dependent data, for instance, some type of reduced frequency representation based on Fourier series or wavelets may be appropriate.

3.4.4.3. Extrapolation. It may first appear that statistical learning can be used when interpolating and integrated learning would be required when extrapolating beyond a training set. There are several difficulties with this belief, however. First, for high-dimensional parameter problems, unless the training data set is sufficiently large, a reduction of the parameter dimension is needed. This process in itself, as stated above, may require an integrated learning approach to define the appropriate reduced parameter space.

Second, we emphasize the definition of extrapolation is not obvious, particularly when one examines the training data in the broader context of the system under study. If the data points are examined from a purely statistical point of view, then extrapolation usually refers to a prediction at a point that lies outside of some region $\boldsymbol{R}$ containing the training data set. There are methods which use ellipsoids or other forms to specify $\boldsymbol{R}$; one minimalist approach is to use a region slightly larger than the convex hull of the training data set. There are some difficulties though: even the convex hull may contain sub-regions that cannot be considered for interpolation. Obvious examples are cases where there are large sub-regions (high-dimensional holes) without training data. Less obvious examples are those where the location of the parameter point has an important impact on system configuration or performance that cannot be directly inferred from the training data (the presence of unexpected phase transitions lying between the

elements of a set of equation of state data, for example). In these latter examples, the training data must be interpreted as more than simply a collection of points in a mathematically structured parameter space.

An important example where apparent interpolation must be analyzed as extrapolation is in complex systems where training data is available only for system components or subsystems, and not for the complete system. For example, in a two-component system, where the learning vector is then naturally partitioned as $\vec{p}_{\mathrm{L}} = (\vec{p}_{\mathrm{L}}^{(1)}, \vec{p}_{\mathrm{L}}^{(2)})$ (in other words, where the parameter space as a *direct sum* of two components), the training data set would have parameter values of the form $(\vec{p}_{\mathrm{L}}^{(1)}, 0)$ or $(0, \vec{p}_{\mathrm{L}}^{(2)})$. This is illustrated in Fig. 5 for the case where $\vec{p}_{\mathrm{L}}^{(1)}$ and $\vec{p}_{\mathrm{L}}^{(2)}$ are one dimensional. While the convex hull of this data set would contain intermediate points of the form $(\vec{p}_{\mathrm{L}}^{(1)}, \vec{p}_{\mathrm{L}}^{(2)})$ as indicated in the figure, in general the response of the system cannot be modeled by interpolating responses of the training data set. The difficulty here is in the interactions between the systems which are not present, and hence are not modeled in the training data set. While the computational code may include one or more models for this interaction, the training data set does not serve as a benchmark for such interaction models. Hence, the apparent interpolation is actually extrapolation, where zero-interaction data is used to predict system behavior with interaction.

The null statistical model is to simply proceed with statistical interpolation, treating the training data such as those in Fig. 5 as points in parameter space. The alternate to this null model is to use domain expertise in conjunction with the computational model to estimate the appropriate types and levels of interaction. When the interactions are complex, a profound level of domain expertise may be needed. The integration of this expertise in a computational learning framework is more difficult than that required in the high-dimensional parameter and comparison problems discussed above.

In summary, computational learning may be used to determine parameter values in a calibration sense, to reduce high-dimensional parameter problems to reduced forms, to quantify and improve comparisons, and for extrapolation.

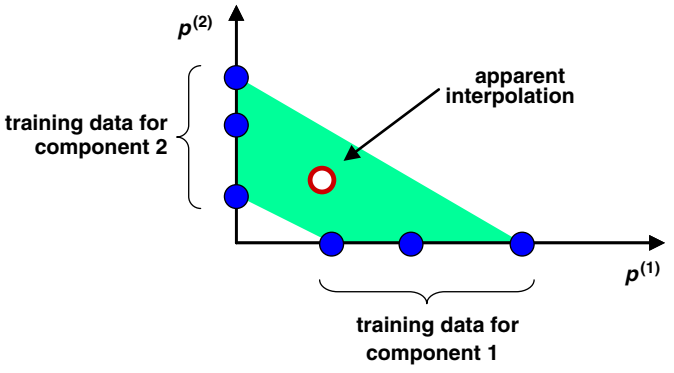

Fig. 5. Illustration of an apparent interpolation point for two-components systems with scalar parameters.

4. Sensitivity analysis

The role of sensitivity analysis in the processes of verification, validation, and calibration is clear. This has been the subject of many previous presentations, for example [69–81]. In all specific instances of the verification, validation and calibration formalism suggested here, understanding the sensitivity of the associated comparisons and credibility quantifications to the individual components in the code parameterizations is fundamental. Parameter sensitivity is also important in guiding our studied reaction to model uncertainty. Parsimony, the reduction of the size of the parameter vector, is guided by sensitivity analysis and remains an important model selection principle. That is, sensitivity analysis is required for understanding the extent to which a model is complicated enough to be credible but not too complicated.

Here, we limit our discussion to two brief comments about the role of sensitivity analysis in V&V that go beyond standard statements about its quantitative importance as found in the above references. First, sensitivity analysis directly contributes to the definition of planned validation activities that culminate in the definition and application of validation benchmarks as defined above. This centers on the use of a Phenomenology Identification and Ranking Table (PIRT) in defining the key requirements of planned validation tasks [82]. The origin of this key idea lies in the application of high-consequence modeling for nuclear reactor safety studies [83], culminating in the development of the Code Scaling, Applicability and Uncertainty (CSAU) methodology. As applied to V&V, sensitivity analysis underlies the determination of the importance, hence priorities, of code elements that must be subjected to V&V in particular studies. The sensitivity analysis may be qualitative and subjective or quantitative and objective for the first specification of a PIRT for V&V. In either case, it is expected that the results of V&V tasks performed in response to the PIRT will create new understanding of the PIRT information and lead to future modifications. This reflects our belief that validation is an ongoing process that can be and should be subject to improvements. Sensitivity analysis supports this concept through iterative improvement of our quantitative understanding of what is important for validation and how priorities should be established at given points in time. In a resource constrained world, this is not only desirable but required.

Second, we stress that from the perspective of prediction, calculation of parametric uncertainties of $M(\vec{p})$, either local or global, suggests the need to predict these sensitivities off the chosen benchmark sets. Ideally, then, we also seek sensitivity benchmarks as part of V&V. In other words, we would like to add sensitivity comparisons, $\boldsymbol{D}_{\mathrm{SEN}}[\cdot,\cdot]$, to the catalog of comparisons presented in Section 2; this

quantity measures the difference between a measure of parameter sensitivity for calculations $S_{\vec{p}}[M(\vec{p}_A, \vec{p}_N)]$ and for benchmarks $S_{\vec{p}}[B(\vec{p}_A)]$. (As one possibility, $S_{\vec{p}}[\cdot]$ could be scaled derivatives of selected calculation or benchmark outputs with respect to the parameter components. It could also be derivable from statistical methods. The referenced work at the beginning of this section illustrates the possibilities.) For example, to apply this concept in validation would require that not only were experimental benchmark data for the benchmark $B(\vec{p}_A)$ be acquired but that experimental *sensitivity* benchmark data be acquired. The degree of agreement with benchmark sensitivity data should then further contribute to the credibility associated with the code through a sensitivity credibility function $\boldsymbol{C}^{\text{red}}_{\text{SEN}}(\boldsymbol{D}_{\text{SEN}}[\cdot,\cdot])$. Given that such a benchmark task can be performed, the overall picture of credibility of the code is given by the accumulated information $\boldsymbol{C}^{\text{red}}_{\text{VER}}(\boldsymbol{D}_{\text{VER}}[\cdot,\cdot])$, $\boldsymbol{C}^{\text{red}}_{\text{VAL}}(\boldsymbol{D}_{\text{VAL}}[\cdot,\cdot])$, and $\boldsymbol{C}^{\text{red}}_{\text{SEN}}(\boldsymbol{D}_{\text{SEN}}[\cdot,\cdot])$.

5. Discussion

This paper presents a discussion intended to highlight the differences between validation and calibration, especially from the view of a rigorous understanding of the (potential) credibility of complex CS&E codes used for particular applications. In particular, we articulated some concepts for formal assessment of credibility of DOE ASC program CS&E codes for use in high-consequence decision applications.

This is a difficult task, and we have presented only a tentative view of the associated problems, certainly not solutions. Our greatest effort over the past few years has been in V&V. This article can be viewed as a summary distillation of some of the insights we have gained regarding calibration as a result of this work. In our own thinking, the prominence and difficulty of verification has also enlarged as we have more deeply engaged validation questions. Verification remains a key problem for validation as well as the broad problem of calibration we sketched in Section 3.

Konikov and Bredehoeft [8] argue that validation is impossible. While made specifically in the context of groundwater flow models, their remarks can be interpreted more generally. Two foci of these authors' argument are problems associated with (1) identification of validation as a goal and (2) a suitable definition of scientific prediction. While we agree that there are great practical difficulties associated with these two issues, we disagree with the substance of the belief in the impossibility of validation.

We believe that V&V are specialized processes that respond to the need to use computational simulation in finite periods of time for key public policy decisions, not simply to reflect the growth of scientific and analytical understanding of complex physical models. Oberkampf, Trucano and Hirsch [6,10] have emphasized this view of V&V. These papers effectively argue at length that V&V is a process of evidence accumulation similar to the formation of legal cases. V&V accumulates evidence to support the case for using codes for specified applications in a manner similar to the way evidence is accumulated in a civil or criminal trial. The processes of high-risk decision making using complex scientific codes as decision-support tools can comfortably work within such a V&V framework, as evidenced by nuclear power licensing methodology requiring reactor safety analyses and predictions [84] and WIPP licensing methodology requiring radionuclide transport analyses and predictions [85]. We have attempted to provide a formalization of the logical nature of this process of evidence accumulation and quantification in this paper, centered on the definition and application of benchmarks.

The drive for prediction is imposed upon the authors of this paper externally through mandates associated with the DOE ASC program. In particular, it is stated in DOE (2004) that the strategic goal is "Predictive simulations and modeling tools, supported by necessary computing resources, to sustain long-term stewardship of the stockpile." The expectation of *predictive modeling* is clear. Whether this is a rational goal is not a point of discussion for this paper. Rather, we must consider what logical elements are both necessary and sufficient to achieve this goal in a way that is useful and understood. Wilson and Boyack [86] and others [24,36–39] argue, and we strongly agree, that the proper context for making high-integrity predictions from computational models has the form *Best Estimate Plus Uncertainty* ($BE + U$).

Our confidence in the quality of BE (Best Estimate) is the accumulation of V&V evidence that we have discussed in this paper. For example, identifying a series of validation benchmarks that are closely related to an intended application of a code and performing a series of calculations with that code that agree well with the benchmarks (in terms of the comparisons of Section 2) lead to increased credibility of the code for the application. This provides a foundation for presenting a computed result as a BE, as well as for understanding the uncertainty in that result. Quantification of U (Uncertainty), which has not been the focus of this paper but is clearly related to the processes and results of V&V, is driven by identification, characterization, and quantification of the uncertainties that appear in the code predictions of BE. The thrust of ($BE + U$) is that prediction is probabilistic precisely because of our inability to complete V&V in some definitive sense and because of uncertainties intrinsic to complex modeling activities. ($BE + U$) recognizes the possibilities of inaccuracy and lack of scientific fidelity in complex simulations that can never be eliminated and that are, therefore, critical for proper communication as part of the process of performing computational studies. The greater the risk of the decision processes that use the modeling results, then the more important it is to work to this paradigm. This information may be very difficult to assemble and communicate. The targeted decision maker may not wish to acknowledge this kind of information, for various reasons. But the overarching goal of prediction as ($BE + U$) remains prominent.

Our interest in the coordination of validation and formal accounting of model uncertainty in calibration methods is rooted in our desire to report (BE + U) in computational studies. Much remains to be done. Konikov and Bredehoeft go so far as to argue that focus on V&V is destructive, that better words if not operating principles for the computational community are "model calibration" and "benchmarking." One of the suggestions we have made in this paper is that these concepts have a formalization that is compatible with V&V as well as dependent on it. In our view, it is more important to focus on V&V, not so much for the reason of glorifying how much we may know about our codes but for the purpose of critical, objective scrutiny of how far they may be from "correct." At the end of the day, this is the element that governs their use and impact on society.

Acknowledgments

The work of the Sandia authors was partially funded by the Applied Mathematical Sciences program, US Department of Energy, Office of Energy Research and performed at Sandia, a multiprogram laboratory operated by Sandia Corporation, a Lockheed-Martin Company, for the US DOE under contract number DE-AC-94AL85000. The work of Igusa was partially funded by the National Science Foundation at the Johns Hopkins University under Grant Number DMI-0087032. The authors would like to thank an anonymous referee and Jon Helton for critically reviewing a preliminary version of this manuscript. We particularly thank the latter reviewer for providing us with a large number of additional references for this paper.

References

[1] United States Department of Energy. Advanced Simulation and Computing Program Plan. Sandia National Laboratories Fiscal Year 2005; Report SAND 2004-4607PP. Issued by Sandia National Laboratories for NNSA's Office of Advanced Simulation & Computing, NA-114, 2004.
[2] United States Department of Defense. DoD Directive No. 5000.59: Modeling and Simulation (M&S) Management, Defense Modeling and Simulation Office, Office of the Director of Defense Research and Engineering, 1994.
[3] United States Department of Defense. Verification, Validation, and Accreditation (VV&A) Recommended Practices Guide, Defense Modeling and Simulation Office, Office of the Director of Defense Research and Engineering, 1996.
[4] AIAA. Guide for the verification and validation of computational fluid dynamics simulations. Reston, VA: American Institute of Aeronautics and Astronautics, AIAA-G-077-1998; 1998.
[5] Trucano TG, Pilch M, Oberkampf WL. On the role of code comparisons in verification and validation. Report SAND2003-2752, Sandia National Laboratories 2003 (Available at http://www.sandia.gov).
[6] Oberkampf WL, Trucano TG. Verification and validation in computational fluid dynamics. Prog Aerospace Sci 2002;38:209–72.
[7] Roache PJ. Verification and validation in computational science and engineering. Albuquerque, NM: Hermosa Publishers; 1998.
[8] Konikov LF, Bredehoeft JD. Groundwater models cannot be validated. Adv Water Resour 1992;15:75–83.
[9] Oreskes N, Shrader-Frechette K, Belitz K. Verification, validation, and confirmation of numerical models in the earth sciences. Science 1994;263:641–6.
[10] Oberkampf WL, Trucano TG, Hirsch C. Verification, validation, and predictive capability in computational engineering and physics. Appl Mech Rev 2004;57(5):345–84.
[11] Beizer B. Software testing techniques. New York: Van Nostrand Reinhold; 1990.
[12] Kaner C, Falk J, Nguyen HQ. Testing computer software. 2nd ed. New York: Wiley; 1999.
[13] Trucano TG, Pilch M, Oberkampf WL. General concepts for experimental validation of ASCI code applications. Report SAND2002-0341, Sandia National Laboratories 2002 (Available at http://www.sandia.gov).
[14] Chen MI, Trucano TG. ALEGRA validation studies for regular, Mach, and double Mach shock reflection in gas dynamics. Report SAND2002-2240, Sandia National Laboratories 2002 (Available at http://www.sandia.gov).
[15] Sandoval DL. CAVEAT calculations of shock interactions. Report LA-11001-MS, Los Alamos National Laboratory 1987.
[16] Trucano TG, et al. Description of the Sandia validation metrics project. Report SAND2001-1339, Sandia National Laboratories 2001 (Available at http://www.sandia.gov).
[17] Stewart TR, Lusk CM. Seven components of judgmental forecasting skill: implications for research and the improvement of forecasts. J Forecasting 1994;13:579–99.
[18] Knupp P, Salari K. Verification of computer codes in computational science and engineering. Boca Raton, FL: Chapman & Hall/CRC; 2003.
[19] Parry GW, Winter PW. Characterization and evaluation of uncertainty in probabilistic risk analysis. Nucl Saf 1981;22(1):28–42.
[20] Paté-Cornell ME. Probability and uncertainty in nuclear safety decisions. Nucl Eng Des 1986;92(2–3):319–27.
[21] Parry GW. On the meaning of probability in probabilistic safety assessment. Reliab Eng Sys Saf 1988;23(4):309–14.
[22] Apostolakis GE. Uncertainty in probabilistic risk assessment. Nucl Eng Des 1989;115:173–9.
[23] Apostolakis GE. The concept of probability in safety assessments of technological systems. Science 1990;250(4986):1359–64.
[24] Helton JC. Treatment of uncertainty in performance assessments of complex systems. Risk Anal 1994;14(4):483–511.
[25] Hoffman FO, Hammonds JS. Propagation of uncertainty in risk assessments: the need to distinguish between uncertainty due to lack of knowledge and uncertainty due to variability. Risk Anal 1994;14(5):707–12.
[26] Helton JC, Burmaster DE. Guest editorial: treatment of aleatory and epistemic uncertainty in performance assessments for complex systems. Reliab Eng Syst Saf 1996;54(2–3):91–4.
[27] Paté-Cornell ME. Uncertainties in risk analysis: six levels of treatment. Reliab Eng Syst Saf 1996;54(2–3):95–111.
[28] Helton JC. Uncertainty and sensitivity analysis in the presence of stochastic and subjective uncertainty. J Stat Comput Simul 1997;57(1–4):3–76.
[29] Helton JC, Oberkampf WL, editor. Special issue: alternative representations of epistemic uncertainty. Reliab Eng Syst Saf 2004;85(1–3):1–376.
[30] Oberkampf WL, DeLand SM, Rutherford BM, Diegert KV, Alvin KF. Error and uncertainty in modeling and simulation. Reliab Eng Syst Saf 2002;75(3):333–57.
[31] Hills RG, Trucano TG. Statistical validation of engineering and scientific models: background. Report SAND99-1256, Sandia National Laboratories 1999 (Available at http://www.sandia.gov).
[32] Zhang R, Mahadevan S. Bayesian methodology for reliability model acceptance. Reliab Eng Syst Saf 2003;80(1):95–103.
[33] Mahadevan S, Rebba R. Validation of reliability computational models using Bayesian networks. Reliab Eng Syst Saf 2005;87:223–32.

[34] Singpurwalla ND, Wilson SP. Statistical methods in software engineering. New York: Springer; 1999.
[35] Campbell K. A brief survey of statistical model calibration ideas. Los Alamos Technical Report LA-UR-02-3157, 2002.
[36] Nichols AL, Zeckhauser RJ. The perils of prudence: how conservative risk assessments distort regulation. Regul Toxicol Pharmacol 1988;8:61–75.
[37] Sielken Jr, RL, Bretzlaff RS, Stevenson DE. Challenges to default assumptions stimulate comprehensive realism as a new tier in quantitative cancer risk assessment. Regul Toxicol Pharmacol 1995;21:270–80.
[38] Caruso MA, et al. An approach for using risk assessment in risk-informed decisions on plant-specific changes to the licensing basis. Reliab Eng Syst Saf 1999;63:231–42.
[39] Paté-Cornell ME. Risk and uncertainty analysis in government safety decisions. Risk Anal 2002;22:633–46.
[40] Oxford English Dictionary. Concise, 11th ed. Oxford: Oxford University Press; 2004.
[41] Red-Horse J, et al. Nondeterministic analysis of mechanical systems. Report SAND2000-0890, Sandia National Laboratories 2000 (Available at http://www.sandia.gov).
[42] Earman J. Bayes or bust? Cambridge, MA: MIT Press; 1992.
[43] Woodward P, Colella P. The numerical simulation of two-dimensional fluid flow with strong shocks. J Comput Phys 1984; 54(1):115–73.
[44] NIST website on statistical process models, regression, parameter estimation, etc.: http://www.itl.nist.gov/div898/handbook/pmd/section1/pmd141.htm
[45] Neter J, Wasserman W, Kuter MH. Applied linear statistical models: regression, analysis of variance, and experimental designs. Homewood, IL: Irwin; 1985.
[46] Bates DM, Watts DG. Nonlinear regression analysis and its applications. New York: Wiley; 1988.
[47] Kennedy MC, O'Hagan A. Bayesian calibration of computer models. J R Stat Soc 2001;63:425–64.
[48] Higdon D, Williams B, Moore L, McKay M, Keller-McNulty S. Uncertainty quantification for combining experimental data and computer simulations. Los Alamos Technical Report, LA-UR 04-6562, 2004.
[49] Bayarri MJ, Berger JO, Higdon D, Kennedy MC, Kotas RP, Sacks J, et al. A framework for validation of computer models. V&V Foundations Conference, 2002.
[50] Cressie NAC. Statistics for spatial data. New York: Wiley; 1993.
[51] Williams C. Gaussian processes. In: Arbib M, editor. The handbook of brain theory and neural networks. Cambridge, MA: MIT Press; 2002.
[52] Gelman AB, Carlin JS, Stern HS, Rubin DB. Bayesian data analysis. Boca Raton, FL: Chapman & Hall/CRC; 1995.
[53] Press SJ. Subjective and objective Bayesian statistics: principles, models, and applications. 2nd ed. New York: Wiley; 2003.
[54] Gilks WR, Richardson S, Spiegelhalter DJ. Markov Chain Monte Carlo in practice. Boca Raton, FL: Chapman and Hall/CRC; 1996.
[55] Gamerman D. Markov Chain Monte Carlo: stochastic simulation for Bayesian inference. Boca Raton, FL: Chapman & Hall/CRC; 1997.
[56] Swiler LP, Trucano TG. Treatment of model uncertainty under calibration. In: Proceedings of the nineth ASCE specialty conference on probabilistic mechanics and structural reliability, 2004.
[57] Campbell K. Exploring Bayesian model calibration: a guide to intuition. Los Alamos Technical Report LA-UR-02-7175, 2002.
[58] Cox DD, Park J-S, Singer CE. A statistical method for tuning a computer code to a data base. Comput Stat Data Anal 2001;37:77–92.
[59] Wolpert DH. The status of supervised learning science circa 1994: the search for a consensus. In: Wolpert DH, editor. The mathematics of generalization. Reading, MA: Addison-Wesley; 1995.
[60] Cucker F, Smale S. On the mathematical foundations of learning. Bull Am Math Soc 2001;39:1–49.
[61] Poggio T, Smale S. The mathematics of learning: dealing with data. Notices Am Math Soc 2003;May:537–44.
[62] Igusa T, Trucano TG. Role of computational learning theory in calibration and prediction. In: Proceedings of the nineth ASCE specialty conference on probabilistic mechanics and structural reliability, 2004.
[63] Hastie T, Tibshirani R, Friedman J. The elements of statistical learning. New York: Springer; 2001.
[64] Cooke RM. Experts in uncertainty: opinion and subjective probability in science. New York: Oxford University Press; 1991.
[65] Meyer MA, Booker JM. Eliciting and analyzing expert judgment: a practical guide. New York: Academic Press; 1991.
[66] Ayyub BM. Elicitation of expert opinions for uncertainty and risks. Boca Raton, FL: CRC Press; 2001.
[67] Johnson RA, Wichern DW. Applied multivariate statistical analysis. Englewood Cliffs, NJ: Prentice-Hall; 1998.
[68] Hardle W. Applied nonparametric regression. New York: Cambridge University Press; 1990.
[69] Beck MB. Water quality modeling: a review of the analysis of uncertainty. Water Resour Res 1987;23:1393–442.
[70] Box GEP, Draper NR. Empirical model-building and response surfaces. New York: Wiley; 1987.
[71] Iman RL. Uncertainty and sensitivity analysis for computer modeling applications. In: Cruse TA, editor. Reliability technology—1992, Winter annual meeting of the American Society of Mechanical Engineers, Anaheim, CA, November 8–13, 1992. New York: American Society of Mechanical Engineers, Aerospace Division, 1992; (28). p. 153–68.
[72] Helton JC. Uncertainty and sensitivity analysis techniques for use in performance safety assessment for radioactive waste disposal. Reliab Eng Syst Saf 1993;42(2–3):327–67.
[73] Blower SM, Dowlatabadi H. Sensitivity and uncertainty analysis of complex models of disease transmission: an HIV model, as an Example. Int Stat Rev 1994;62(2):229–43.
[74] Hamby DM. A review of techniques for parameter sensitivity analysis of environmental models. Environ Monit Assess 1994;32(2):135–54.
[75] Saltelli A, Scott M. Guest editorial: the role of sensitivity analysis in the corroboration of models and its link to model structural and parametric uncertainty. Reliab Eng Syst Saf 1997;57:1–4.
[76] Kleijnen JPC, Helton JC. Statistical analyses of scatterplots to identify important factors in large-scale simulations, 1: review and comparisons of techniques. Reliab Eng Syst Saf 1999;65(2):147–85.
[77] Saltelli A, Chan K, Scott EM. Sensitivity analysis. New York: Wiley; 2000.
[78] Frey HC, Patil SR. Identification and review of sensitivity analysis methods. Risk Anal 2002;22(3):553–78.
[79] Helton JC, Davis FJ. Illustration of sampling-based methods for uncertainty and uncertainty analysis. Risk Anal 2002;22(3):591–622.
[80] Ionescu-Bujor M, Cacuci DG. A comparative review of sensitivity and uncertainty analysis of large-scale systems—I: deterministic methods. Nucl Sci Eng 2004;147(3):189–203.
[81] Cacuci DG, Ionescu-Bujor MA. Comparative review of sensitivity and uncertainty analysis of large-scale systems—II: statistical methods. Nucl Sci Eng 2004;147(3):204–17.
[82] Pilch M, et al. Guidelines for Sandia ASCI verification and validation plans—content and format: version 2.0. Report SAND2000-3101, Sandia National Laboratories 2001 (Available at http://www.sandia.gov).
[83] Boyack BE. Quantifying reactor safety margins. Part 1: an overview of the code scaling, applicability, and uncertainty evaluation methodology. Nucl Eng Des 1990;119:1–15.
[84] Breeding RJ, Helton JC, Gorham ED, Harper FT. Summary description of the methods used in the probabilistic risk assessments for NUREG-1150. Nucl Eng Des 1992;135(1):1–27.
[85] Helton JC, Anderson DR, Basabilvazo G, Jow H-N, Marietta MG. Conceptual structure of the 1996 performance assessment for the waste isolation pilot plant. Reliab Eng Syst Saf 2000;69(1–3):151–65.
[86] Wilson GE, Boyack BE. The role of the PIRT process in experiments, code development and code applications associated with reactor safety analysis. Nucl Eng Des 1998;186:23–37.