Verification, validation, and predictive capability in computational engineering and physics

William L Oberkampf

Validation and Uncertainty Estimation Department, MS 0828, Sandia National Laboratories, PO Box 5800, Albuquerque NM 87185-0828; wloberk@sandia.gov

Timothy G Trucano

Optimization and Uncertainty Estimation Department, Sandia National Laboratories, Albuquerque NM; tgtruca@sandia.gov

Charles Hirsch

Department of Fluid Mechanics, Vrije Universiteit Brussel, Brussels, Belgium; hirsch@stro.vub.ac.be

Developers of computer codes, analysts who use the codes, and decision makers who rely on the results of the analyses face a critical question: How should confidence in modeling and simulation be critically assessed? Verification and validation (V&V) of computational simulations are the primary methods for building and quantifying this confidence. Briefly, verification is the assessment of the accuracy of the solution to a computational model. Validation is the assessment of the accuracy of a computational simulation by comparison with experimental data. In verification, the relationship of the simulation to the real world is not an issue. In validation, the relationship between computation and the real world, ie, experimental data, *is* the issue. This paper presents our viewpoint of the state of the art in V&V in computational physics. (In this paper we refer to all fields of computational engineering and physics, eg, computational fluid dynamics, computational solid mechanics, structural dynamics, shock wave physics, computational chemistry, etc, as computational physics.) We describe our view of the framework in which predictive capability relies on V&V, as well as other factors that affect predictive capability. Our opinions about the research needs and management issues in V&V are very practical: What methods and techniques need to be developed and what changes in the views of management need to occur to increase the usefulness, reliability, and impact of computational physics for decision making about engineering systems? We review the state of the art in V&V over a wide range of topics, for example, prioritization of V&V activities using the Phenomena Identification and Ranking Table (PIRT), code verification, software quality assurance (SQA), numerical error estimation, hierarchical experiments for validation, characteristics of validation experiments, the need to perform nondeterministic computational simulations in comparisons with experimental data, and validation metrics. We then provide an extensive discussion of V&V research and implementation issues that we believe must be addressed for V&V to be more effective in improving confidence in computational predictive capability. Some of the research topics addressed are development of improved procedures for the use of the PIRT for prioritizing V&V activities, the method of manufactured solutions for code verification, development and use of hierarchical validation diagrams, and the construction and use of validation metrics incorporating statistical measures. Some of the implementation topics addressed are the needed management initiatives to better align and team computationalists and experimentalists in conducting validation activities, the perspective of commercial software companies, the key role of analysts and decision makers as code customers, obstacles to the improved effectiveness of V&V, effects of cost and schedule constraints on practical applications in industrial settings, and the role of engineering standards committees in documenting best practices for V&V. There are 207 references cited in this review article. [DOI: 10.1115/1.1767847]

Transmitted by Associate Editor JN Reddy

1 INTRODUCTION

1.1 Background

During the last three or four decades, computer simulations of physical processes have been increasingly used in scientific research and in the analysis and design of engineered systems. The systems of interest have been existing or proposed systems that operate, for example, at design conditions, off-design conditions, and failure-mode conditions in accident scenarios. The systems of interest have also been natural systems, for example, computer simulations for environmental impact, as in the analysis of surface-water quality and the risk assessment of underground nuclear-waste repositories. These kinds of predictions are beneficial in the development of public policy, the preparation of safety procedures, and the determination of legal liability. Thus, because of the impact that modeling and simulation predictions can have, the credibility of the computational results is of great concern to engineering designers and managers, public officials, and those who are affected by the decisions that are based on these predictions.

For engineered systems, terminology such as ''virtual prototyping'' and ''virtual testing'' is now being used in engineering development to describe numerical simulation for the design, evaluation, and ''testing'' of new hardware and even entire systems. This new trend of modeling and simulation-based design is primarily driven by increased competition in many markets, eg, aircraft, automobiles, propulsion systems, and consumer products, where the need to decrease the time and cost of bringing products to market is intense. This new trend is also driven by the high cost and time that are required for testing laboratory or field components as well as complete systems. Furthermore, the safety aspects of the product or system represent an important, sometimes dominant, element of testing or validating numerical simulations. The potential legal and liability costs of hardware failures can be staggering to a company, the environment, or the public. This consideration is especially critical, given that we may be interested in the reliability, robustness, or safety of high-consequence systems that cannot ever be physically tested. Examples are the catastrophic failure of a full-scale containment building for a nuclear power plant, a fire spreading through (or explosive damage to) a high-rise office building, ballistic missile defense systems, and a nuclear weapon involved in a transportation accident. In contrast, however, an inaccurate or misleading numerical simulation for a scientific research project has comparatively no impact.

Users and developers of computational simulations today face a critical question: How should confidence in modeling and simulation be critically assessed? Verification and validation (V&V) of computational simulations are the primary methods for building and quantifying this confidence. Briefly, verification is the assessment of the accuracy of the solution to a computational model, primarily by comparison with known solutions. Validation is the assessment of the accuracy of a computational simulation by comparison with experimental data. In verification, the relationship of the simulation to the real world is not an issue. In validation, the relationship between computation and the real world, ie, experimental data, *is* the issue.

In the United States, the Defense Modeling and Simulation Office (DMSO) of the Department of Defense (DoD) has been the leader in the development of fundamental concepts and terminology for V&V [1,2]. In the past five years, the Accelerated Strategic Computing Initiative (ASCI) of the Department of Energy (DOE) has also taken a strong interest in V&V. The ASCI program is focused on computational physics and computational mechanics, whereas the DMSO has traditionally emphasized high-level systems engineering, such as ballistic missile defense systems, warfare modeling, and simulation-based system acquisition. Of the work conducted by DMSO, Cohen *et al* recently observed [3]: ''Given the critical importance of model validation..., it is surprising that the constituent parts are not provided in the (DoD) directive concerning... validation. A statistical perspective is almost entirely missing in these directives.'' We believe this observation properly reflects the state of the art in V&V, not just the directives of DMSO. That is, the state of the art has not developed to the place where one can clearly point out all of the actual methods, procedures, and process steps that *must* be undertaken for V&V.

It is fair to say that computationalists (code users and code developers) and experimentalists in the field of fluid dynamics have been pioneers in the development of methodology and procedures in validation. However, it is also fair to say that the field of computational fluid dynamics (CFD) has, in general, proceeded along a path that is largely independent of validation. There are diverse reasons why the CFD community has not perceived a strong need for code V&V, especially validation. One reason is that a competitive and frequently adversarial relationship (at least in the United States) has often existed between computationalists and experimentalists, resulting in a lack of cooperation between the two groups. We, on the other hand, view computational simulation and experimental investigations as complementary and synergistic. To those who might say, ''Isn't that obvious?'' we would answer, ''It should be, but they have not always been viewed as complementary.'' In retrospect, the relationship between computationalists and experimentalists is probably understandable because it represents the classic case of a new technology (computational simulation) that is rapidly growing and attracting a great deal of visibility and funding support that had been in the domain of the older technology (experimentation).

It is our view that the field of structural dynamics has enjoyed, in general, a more beneficial and synergistic relationship between computationalists and experimentalists. We believe this type of relationship has developed because of the strong dependence of structural dynamics models on experimental measurements. Most researchers in the field of structural dynamics have referred to this interaction as ''model validation.'' As discussed in Section 2.1, we believe a more precise term for this interaction is either ''model updating''

or ''model calibration.'' That is, the primary interaction between computation and experiment is to update or ''tune'' the unknown parameters in the computational model using the experimental results from modal testing. This approach in structural dynamics has proven to be very effective because it permits the estimation of specific constituents of poorly known physics in the computational models. In structural dynamics, the problem primarily arises in poor understanding of the localized deformation of connectors and joints between structural elements in the computational models. A similar approach is used in fluid dynamics when dealing with turbulent reacting flows and two-phase flows.

From a historical perspective, the operations research (OR) and systems engineering communities have provided the philosophical foundations for verification and validation. With the recent interest in V&V from the CFD and computational physics communities, one recognizes significant differences in perspectives between the historical view and the view held by the computational physics community. (For simplicity, we will refer to all fields of computational engineering and physics, eg, CFD, computational solid mechanics, structural dynamics, shock wave physics, computational chemistry, etc, as *computational physics*.)

1.2 Basic terminology and methodology

There is a wide variety of different meanings used for V&V in the various technical disciplines. For example, the meanings used by the Institute of Electrical and Electronics Engineers (IEEE) and the software quality assurance community are different from the meanings used in the DoD modeling and simulation community. And given that members of the different technical communities often work together on V&V activities, we expect there will be long-term ambiguity and confusion resulting from terminology differences. Although we have not reviewed all of the different meanings in this paper, we refer the reader to references that describe the varying usage [2,4–9]. For reviews of the historical development of the terminology for verification, validation, and prediction see, for example, [10–12].

The DMSO under the DoD has played a major role in attempting to standardize the definitions of V&V. In 1994 the DoD published definitions of V&V that are clear, concise, and directly useful by themselves [1,2,8]. From the perspective of the computational engineering and physics communities, however, the definition of verification by the DoD does not make it clear that the accuracy of the numerical solution to the partial differential equations (PDEs) should be included in the definition. To clarify this issue, the CFD Committee on Standards of the American Institute of Aeronautics and Astronautics (AIAA) proposed a slight modification to the DoD definition. This paper will use the DoD definitions, with the AIAA modification for verification [9].

Verification: 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: 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.

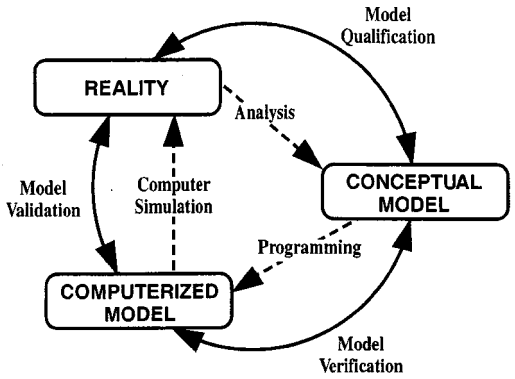


Fig. 1 Phases of modeling and simulation and the role of V&V [13]

We think an excellent graphical representation of the fundamental meaning of V&V was constructed by the Society for Computer Simulation (SCS) in 1979, as shown in Fig. 1. The figure identifies two types of models: a conceptual model and a computerized model. The conceptual model is composed of all mathematical modeling data and mathematical equations that describe the physical system or process of interest. The conceptual model is produced by analyzing and observing the physical system of interest. In key applications of computational physics, the conceptual model is dominated by the PDEs for conservation of mass, momentum, and energy. The conceptual model also includes all of the auxiliary equations, such as turbulence models, constitutive models for materials, and electromagnetic cross-section models, and all of the initial conditions and boundary conditions of the PDEs. The computerized model is an operational computer program that implements a conceptual model. Modern terminology usually refers to the conceptual model as the mathematical model, and the computerized model as the computer model or code. Figure 1 clearly shows that verification deals with the fidelity between the conceptual model and the computerized model and that validation clearly deals with the fidelity between the computerized model and experimental measurements. The SCS defined *qualification* as the ''Determination of adequacy of the conceptual model to provide an acceptable level of agreement for the domain of intended application.'' According to this definition, qualification would deal with issues such as definition of the system of interest, effects of the environment on the system, and the choice of PDEs in computational physics. The topic of conceptual model qualification will not be addressed in this paper.

Fundamentally, V&V are tools for assessing the accuracy of the conceptual and computerized models. For much of the OR work, the assessment was so difficult, if not impossible, that V&V became more associated with the issue of credibility, ie, the quality, capability, or power to elicit belief. In science and engineering, however, quantitative assessment of accuracy, at least for some important physical cases, is mandatory; it is a necessary condition for credibility. And in cer-

tain situations, assessment can only be conducted using subscale physical models or a subset of the active physical processes. Regardless of the difficulties and constraints, methods must be devised for measuring the accuracy of the model for as many application conditions of the model as is deemed appropriate. As the complexity of a model increases, its accuracy and range of applicability become more difficult to assess.

Some important implications and subtleties in the definitions of V&V need to be addressed at this point. The first significant feature is that both V&V are ''processes of determining.'' That is, they are ongoing activities that do not have a clearly defined completion point. Completion or sufficiency is usually determined by practical issues such as budgetary constraints and intended uses of the model. The definitions include the ongoing nature of the process because of an unavoidable but distressing fact: the correctness and accuracy of a computerized, or computational, model cannot be demonstrated for all possible applications and code options, except for trivial models. Trivial models are clearly not of interest. All-encompassing proofs of correctness, such as those developed in mathematical analysis and logic, do not exist in complex modeling and simulation. Indeed, one cannot prove that complex computer codes have no errors. Likewise, models of physics cannot be proven correct; they can only be disproved. Thus, V&V activities can only assess the correctness or accuracy of the specific cases tested.

The emphasis on ''accuracy'' is the second feature that is common in the definitions of V&V. This feature assumes that a measure of correctness can be determined. In verification activities, accuracy is generally measured in relation to benchmark solutions of simplified, or closely related, model problems. Benchmark solutions refer either to analytical solutions, ie, exact solutions to the PDEs with the specified initial conditions and boundary conditions, or to highly accurate numerical solutions. However, we believe that in the solution of nonlinear PDEs or solutions with discontinuities or singularities, the most reliable benchmark solutions are analytical solutions. In validation activities, accuracy is measured in relation to experimental data, ie, our best indication of reality. Since all experimental data have random (statistical) and bias (systematic) errors, the issue of ''correctness,'' in an absolute sense, becomes impossible. From an engineering perspective, however, we do not require ''absolute truth:'' we only expect a *statistically meaningful comparison* of computational results and experimental measurements for the specific system responses assessed. These issues are discussed in more detail in Section 2.4.

Effectively, verification provides evidence (substantiation) that the conceptual (continuum mathematics) model is solved correctly by the discrete-mathematics computer code. (Note: When we refer to ''continuum mathematics,'' we are *not* referring to the physics being modeled by the mathematics. For example, the equations for noncontinuum fluid dynamics are commonly expressed with continuum mathematics.) Verification does not address whether the conceptual model has any relationship to the real world. Validation, on the other hand, provides evidence (substantiation) for how accurately

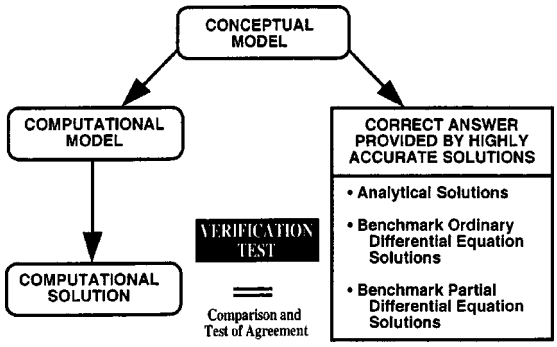


Fig. 2 Verification process [9]

the computational model simulates the real world for system responses of interest. This perspective implies that the model is solved accurately. However, multiple errors or inaccuracies can cancel one another and give the appearance of a validated solution. Verification, thus, is the first step of the validation process and, while not simple, is much less involved than the more complex statistical nature of validation. Validation addresses the question of the fidelity of the model to specific conditions of the real world. As Roache [14] succinctly states, ''Verification deals with mathematics; validation deals with physics.''

As a final comment on terminology, it is our view that the DoD definition of validation does *not* include the concept of *adequacy of the computational result for the intended uses* of the model. Stated differently, we argue that validation is the process of determining the degree to which computational simulation results agree with experimental data. We recognize that this interpretation of the meaning of validation is narrower than the interpretation that is widely accepted in the DoD community. It is our understanding that the DoD community considers validation to be the process of determining the degree to which the computational model results are *adequate* for the application of interest. This important topic of divergent interpretations of validation is briefly discussed in recommendations for future work, Section 4.4. Regardless of whether the reader agrees with our interpretation, we have chosen to clarify our view now to help avoid confusion throughout the paper. Stating our view succinctly: validation deals with quantified comparisons between experimental data and computational data; *not* the adequacy of the comparisons.

In 1998 the Computational Fluid Dynamics Committee on Standards of the AIAA contributed to the basic methodology and procedures for V&V [9]. The *Guide for the Verification and Validation of Computational Fluid Dynamics Simulations*, referred to herein as the ''AIAA Guide,'' was the first engineering standards document that addressed issues of particular concern to the computational physics community. In the following paragraphs we have briefly summarized the basic methodology for V&V from the AIAA Guide.

The fundamental strategy of verification is to identify, quantify, and reduce errors in the computational model and its numerical solution. Figure 2 depicts the verification process of comparing the numerical solution from the code in

question with various types of highly accurate solutions [9]. Given a numerical procedure that is stable, consistent, and robust, the five primary sources of errors in computational physics solutions are: insufficient spatial discretization, insufficient temporal discretization, insufficient iterative convergence, computer round-off, and computer programming. The emphasis in verification is on identifying and quantifying errors from these various sources, as well as on demonstrating the stability, consistency, and robustness of the numerical scheme. Stated differently, an analytical or formal error analysis is inadequate in the verification process; verification relies on *demonstration* and *quantification* of numerical accuracy. Note that the recommended methodology presented here applies to finite-difference, finite-volume, finite-element, and boundary-element discretization procedures.

The first three error sources listed in the previous paragraph (spatial discretization, temporal discretization, and iterative procedure) are considered to be within the traditional realm of computational physics and there is extensive literature dealing with each of these topics. The fourth error source, computer round-off, is rarely dealt with in computational physics. Collectively, these four topics in verification could be referred to as solution verification or solution error assessment. The fifth error source, computer programming, is generally considered to be in the realm of computer science or software engineering. Programming errors, which can occur, for example, in input data files, source-code programming of the numerical algorithm, output data files, compilers, and operating systems, generally are addressed using methods and tools in software quality assurance (SQA), also referred to as software quality engineering (SQE) [15]. The identification of programming errors is usually referred to as code verification, as opposed to solution verification [14,16]. In our opinion, the perspectives and emphases of SQA and computational physics are sufficiently different and thus should be addressed separately: code verification, which emphasizes the accuracy of the numerical algorithm that is used to solve the discrete form of the PDEs; and SQA, which emphasizes the issues in computer science. Each of these topics is discussed in Section 2.3.

The fundamental strategy of validation involves identifying and quantifying the error and uncertainty in the conceptual and computational models, quantifying the numerical error in the computational solution, estimating the experimental uncertainty, and then comparing the computational results with the experimental data. This strategy *does not* assume that the experimental measurements are more accurate than the computational results. The strategy only asserts that experimental measurements are the most faithful reflections of reality for the purposes of validation. Validation requires that the estimation process for error and uncertainty must occur on both sides of the coin: mathematical physics and experiment. Figure 3 depicts the validation process of comparing the computational results of the modeling and simulation process with experimental data from various sources [9].

Because of the infeasibility and impracticality of conducting true validation experiments on most complex or large

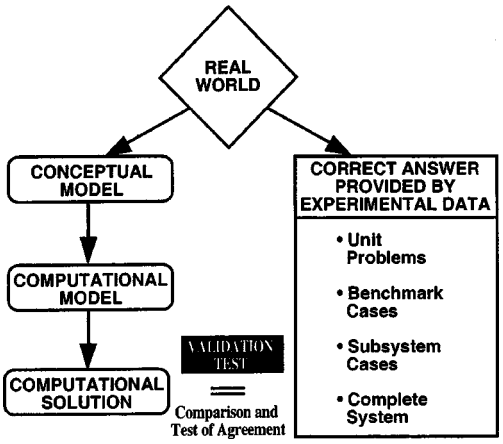


Fig. 3 Validation process [9]

scale systems, the recommended method is to use a building-block approach [9,17–21]. This approach divides the complex engineering system of interest into at least three progressively simpler tiers: subsystem cases, benchmark cases, and unit problems. The strategy in the tiered approach is to assess how accurately the computational results compare with the experimental data (with quantified uncertainty estimates) at multiple degrees of physics coupling and geometric complexity. The approach is clearly constructive in that it 1) recognizes that there is a hierarchy of complexity in systems and simulations and 2) recognizes that the quantity and accuracy of information that is obtained from experiments vary radically over the range of tiers. Furthermore, this approach demonstrates that validation experiments can be conducted at many different levels of physics and system complexity. Each comparison of computational results with experimental data allows an inference of validation concerning tiers both above and below the tier where the comparison is made. However, the quality of the inference depends greatly on the complexity of the tiers above and below the comparison tier. For simple physics, the inference may be very strong, eg, laminar, single phase, Newtonian, nonreacting flow, and rigid-body structural dynamics. However, for complex physics, the inference is commonly very weak, eg, turbulent reacting flow and fracture dynamics. This directly reflects the quality of our scientific knowledge about the experiments and calculations that are being compared for more complex tiers.

1.3 Outline of the paper

This paper presents our viewpoint on the state of the art in V&V in computational physics. We have not attempted herein to provide a comprehensive review of the multitudinous contributions to V&V from many diverse fields. This literature represents distinctively different perspectives and approaches, ranging from engineering and physics to operations research. Recent reviews of the literature are given in [12,14,22–27]. Recent work providing wide-ranging procedures in V&V is described in [12,28–30]. We have attempted in this paper to bring together many different perspectives on V&V, highlight those perspectives that are

effective from a practical engineering viewpoint, suggest future research topics, and discuss key implementation issues that are necessary to improve the effectiveness of V&V. Our views about the research needs and management issues in V&V are very practical: What methods and techniques need to be developed and what changes in the views of management need to occur to increase the usefulness, reliability, and impact of computational physics for decision making about engineering systems?

Section 2 describes the primary processes in V&V and the relationship of V&V to predictive capability. V&V are the key building blocks in assessing the confidence in the predictive capability of a computational physics code. The section begins with a description of the framework in which predictive capability relies on V&V, as well as other factors that affect predictive capability. We also briefly discuss how this framework is related to more traditional approaches of predictive capability, eg, Bayesian estimation. The importance of requirements for a computational-physics capability is stressed so that effective and efficient V&V activities can be conducted. Following the framework discussion, we present a summary of verification activities, emphasizing code verification, SQA, and numerical error estimation. A discussion of validation activities follows, highlighting methods for focusing on the validation experiments most important to the predictive capability required. The section concludes with a discussion of hierarchical experiments for validation, characteristics of validation experiments, the need to perform nondeterministic simulations in comparisons with experimental data, and validation metrics.

Section 3 discusses the research issues that we believe must be addressed for V&V to be more effective in improving confidence in computational predictive capability. We begin with a discussion of methods for prioritizing V&V assessment activities. Needed research in verification activities, such as statistical methods for code verification and the method of manufactured solutions, are discussed. Topics discussed with regard to validation research are: development and use of hierarchical validation diagrams; and the construction and use of validation metrics incorporating statistical measures. We close the section with the difficult research issue of how to quantify uncertainty in predictive capability, given an arbitrary set of validation experiments, eg, how to estimate the uncertainty of a computational prediction when, in some sense, we are extrapolating beyond the validation database.

Section 4 discusses issues related to improving the implementation of V&V activities in a realistic engineering environment and a commercial software environment. These issues include needed improvements in management of verification activities, the key role of analysts and decision makers as code customers, and obstacles to the improved effectiveness of V&V. Examples of obstacles are the conflicting perspectives of code developers, analysts, hardware designers, and experimentalists; competition between organizations and nations; and the loss of focus on the needs of the customer for computational physics analyses. Also discussed are the effects of cost and schedule constraints on practical applications in industrial settings, the demands of customers on commercial software companies, and the large-scale validation database activity underway in Europe.

Section 5 presents some closing remarks concerning the status of V&V and the need for improvement.

2 PRIMARY PROCESSES

2.1 Framework for predictive capability

The issues underlying the V&V of mathematical and computational models of physical systems, including those systems with strong human interaction, touch on the very foundations of mathematics, science, and human behavior. Verification is rooted in issues pertaining to continuum and discrete mathematics and to the accuracy and correctness of complex logical structures (computer codes). Validation is deeply rooted in the question of how formal constructs (mathematical models) of nature and human behavior can be tested by physical observation. In the OR field, the systems being analyzed can be extraordinarily complex, such as industrial planning models, marketing models, national and world economic models, monetary investment models, and military conflict models. For these types of situations, one must deal with statistical models where statistical calibration and parameter estimation are crucial elements in building the models. These complex models commonly involve a strong coupling of complex physical processes, human behavior, and computer-controlled systems. For such complex systems and processes, fundamental conceptual issues immediately arise about how to assess the accuracy of the model and the resulting simulations. Indeed, the predictive accuracy of most of these models cannot be assessed in any meaningful way, except for predictive cases that are very near, in some sense, the calibration database.

In the AIAA Guide, prediction is defined as ''use of a computational model to foretell the state of a physical system under conditions for which the computational model has not been validated.'' A prediction refers to the computational simulation of a specific case of interest that is *different* from cases that have been validated. This definition of prediction is more restrictive than the general scientific meaning of prediction because it eliminates past comparisons of computational results with experimental data. This definition segregates the general meaning of prediction and only refers to prediction, not retrodiction (replication of previously obtained results). If this restriction is not made, then one is only demonstrating previous agreement with experimental data in the validation database. The results of the process of validation should be viewed as historical statements. Thus, the validation database represents reproducible evidence that a model has achieved a given level of accuracy in the solution of specified problems. From this perspective, it becomes clear that validation comparisons do not directly allow one to make claims about the accuracy of predictions: they allow inferences to be made. The strength of the inferences depends on many factors. The suggested relationship between validation and prediction is shown in Fig. 4.

Figure 4 attempts to capture the distinction between vali-

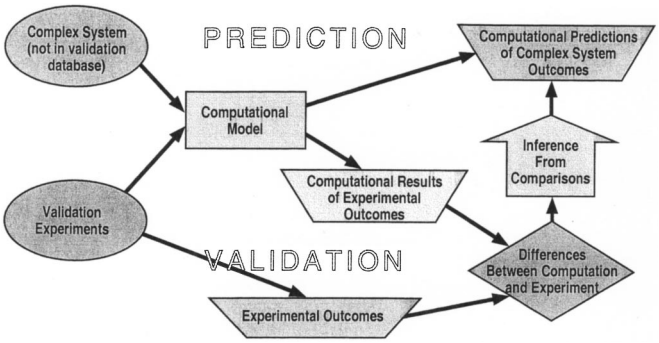


Fig. 4 Relationship of validation to prediction [12,31]

dation and prediction. The bottom portion of the figure represents the validation process. Although it is not readily apparent, the validation process in Fig. 4 is fundamentally the same as that shown in Fig. 3. In Fig. 4, the block ''Validation Experiments'' produces one or more realizations of the ''real world.'' The ''Experimental Outcomes'' are the physical realizations, ie, the experimental data from the experiment. The physical conditions from the actual validation experiments, ie, model input parameters, initial conditions, and boundary conditions, are input to the ''Computational Model,'' which produces the ''Computational Results of Experimental Outcomes.'' These results are then compared with the experimentally determined outcomes in the block ''Differences Between Computation and Experiment.'' Based on the magnitude of these differences in quantities of interest in the simulation and on the understanding of the physical process, an ''Inference from Comparisons'' is made.

The upper portion of Fig. 4 represents the prediction process. The ''Complex System'' of interest should drive the entire modeling and simulation process, but most of the realizations of interest, ie, predictions, are not in the validation database. That is, when a physical realization is conducted as part of the validation database, regardless of the tier as discussed in Section 2.4.2, the realization becomes part of the ''Validation Experiments.'' Predictions for conditions of interest are made using the ''Computational Model,'' resulting in ''Computational Predictions of Complex System Outcomes.'' The confidence in these predictions is determined by the ''Inference from Comparisons'' and the level of understanding of the physical process.

The process of logical inference of accuracy of a computational model stemming from its associated validation database is analogous to similar processes and conclusions for classical scientific theories. However, we argue that the strength or confidence in the inference from computational simulation is, and should be, much weaker than traditional scientific theories. Computational simulation relies on the same logic as traditional science, but it also relies on many additional mathematical issues, eg, discretization algorithms and grid quality, and practical implementation issues, eg, computer hardware, operating-system software, source-code reliability, and analyst skill, that are not present in classical scientific theories. (In this paper ''analyst'' refers to the individual, or group of individuals, who constructs the computational model, including all physical and numerical inputs to the model, and then uses the computer code to produce a simulation.) One of the key theoretical issues is the state of knowledge of the process being modeled. Zeigler, *et al* [27] gives a detailed discussion of hierarchical levels of knowledge of a system. For physical processes that are well understood both physically and mathematically, the inference can be quite strong. For complex physical processes, the inference can be quite weak. A general mathematical method for determining how the value and consequence of such an inference degrades as the physical process becomes more complex has not been formulated. For example, in a complex physical process how do you determine ''how nearby'' the prediction case is from cases in the validation database? This is a topological question applied to an ill-defined space. Struggling with the strength or quantification of the inference in a prediction is, and will remain, an important topic of research [27,32].

To better understand the present perspective of the relationship of prediction to validation, consider Fig. 5. The horizontal axis of the figure is labeled *system or environmental parameters*. These are parameters in the model of a physical system that typically come from the initial conditions or boundary conditions. Examples of these parameters are: initial speed of an automobile in a crash environment, Mach

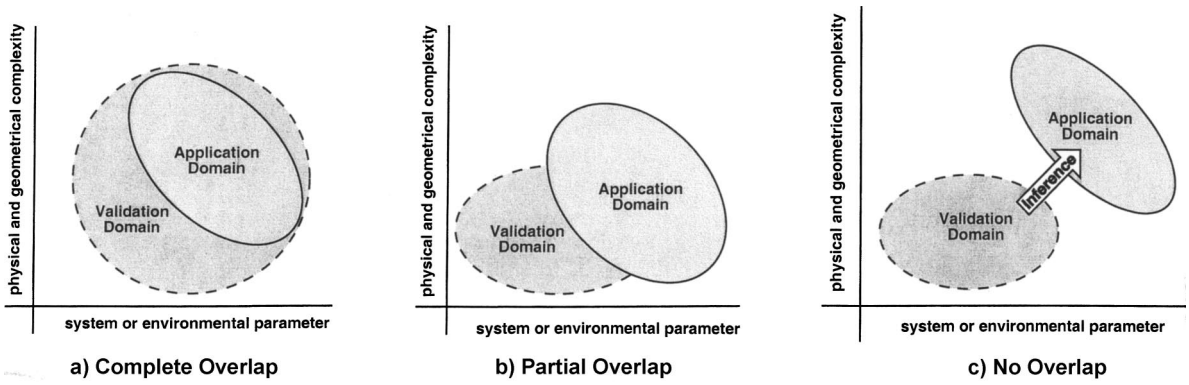


Fig. 5 Possible relationships of the validation domain to the application domain

number in a gas dynamics problem, amplitude or frequency of a vibrational excitation, and magnitude of the environmental heat flux in a heat transfer problem. The vertical axis is a metavariable that we refer to as *physical and geometrical complexity*. This axis, although difficult to quantify, could be the hierarchy of physical and geometrical complexity as shown in Fig. 3: unit problem, benchmark case, subsystem case, or complete system. A point along this axis would represent a specified physical system or process that undergoes a change as a function of the system or environmental parameter. This type of diagram was recently suggested by Easterling [31].

The ''validation domain'' in Fig. 5 suggests three features. First, in this region we have high confidence that the relevant physics is understood and modeled at a level that is commensurate with the needs of the application. Second, this confidence has been quantitatively demonstrated by satisfactory agreement between computations and experiments in the validation database for some range of applicable parameters in the model. And third, the boundary of the domain indicates that outside this region there is a degradation in confidence in the quantitative predictive capability of the model. Stated differently, outside the validation domain the model may be credible, but its quantitative capability has not been demonstrated. The ''application domain'' indicates the region where predictive capability is needed from the model for the applications of interest.

Figure 5*a* depicts the prevalent situation in engineering, that is, the complete overlap of the validation domain with the application domain. The vast majority of modern engineering system design is represented in Fig. 5*a*. Figure 5*b* represents the occasional engineering situation where there is significant overlap between the validation domain and the application domain. Some examples are: the prediction of crash response of new automobile structures, entry of spacecraft probes into the atmosphere of another planet, and the structural response of new designs for deep-water offshore oil platforms. Figure 5*c* depicts the situation where there is no overlap between the validation domain and the application domain. We believe that predictions for many high-consequence systems are in this realm because we are not able to perform experiments for closely related conditions. We believe that the inference from the validation domain suggested in Fig. 5*c* can only be made using *both* physics-based models and statistical methods. The need to perform this extrapolation reinforces our need for models to be judged on the basis of achieving the right answers for the right reasons in the validation regime. Model calibration, which employs explicit tuning or updating of model parameters to achieve improved agreement with existing validation experiments, does not fully assess uncertainty in the predictive use of the model. Stated more strongly, it is not convincing that model calibration provides a starting point for the inference process demanded in Fig. 5*c*.

As a final conceptual issue concerning predictive capability, we strongly believe that a more careful distinction between error and uncertainty is needed than has traditionally been made in the computational physics literature. We define *error* based on the AIAA Guide [9] (see also [33–35]) to be ''A recognizable deficiency in any phase or activity of modeling and simulations that is not due to lack of knowledge.'' This definition emphasizes the required feature that the deficiency is identifiable or knowable upon examination, that is, the deficiency is not fundamentally caused by lack of knowledge. This definition leads to the so-called *acknowledged* and *unacknowledged* errors. An acknowledged error is characterized by knowledge of divergence from an approach or ideal condition that is considered to be a baseline for accuracy. Examples of acknowledged errors are finite precision arithmetic in a computer, approximations made to simplify the modeling of a physical process, and conversion of PDEs into discrete equations. An acknowledged error can therefore, in principle, be measured because its origins are fully identified. Solution verification, ie, solution error estimation, deals with acknowledged errors. For example, we know that the discretization of the PDE and the approximate solution of the discrete equations introduce acknowledged errors. Unacknowledged errors are blunders or mistakes, such as programming errors, input data errors, and compiler errors. There are no straightforward methods for estimating, bounding, or ordering the contributions of unacknowledged errors. Code verification and SQA activities primarily deal with unacknowledged errors, which is another important reason these activities are essential.

In a technical sense, the term uncertainty seems to have two rather different meanings. The first meaning of uncertainty has its roots in probability and statistics: the estimated amount or percentage by which an observed or calculated value may differ from the true value. This meaning of uncertainty has proven its usefulness over many decades, particularly in the estimation of random uncertainty in experimental measurements [36]. The second meaning of uncertainty relates to lack of knowledge about physical systems, particularly in the prediction of future events and the estimation of system reliability. The probabilistic-risk and safety assessment communities, as well as the reliability engineering community, use the term *epistemic uncertainty* in this latter sense. The risk assessment community [37–42] refers to the former meaning, random uncertainty, as *aleatory uncertainty*, as does the information theory community [43–48].

Aleatory uncertainty is used to describe the *inherent variation* associated with the physical system or environment being considered. Sources of aleatory uncertainty can commonly be singled out from other contributors to uncertainty by their representation as randomly distributed quantities that can take on values in an established or known range, but for which the exact value will vary by chance from unit to unit or from time to time. The mathematical representation most commonly used to characterize aleatory uncertainty is a probability distribution. Aleatory uncertainty is also referred to in the literature as variability, irreducible uncertainty, inherent uncertainty, and stochastic uncertainty.

Epistemic uncertainty as a cause of nondeterministic behavior derives from some level of *ignorance* or *lack of knowledge* about the system or the environment. Thus, an increase in knowledge or information can lead to a reduction in the predicted uncertainty of the system's response—all things being equal. Epistemic uncertainty can be introduced

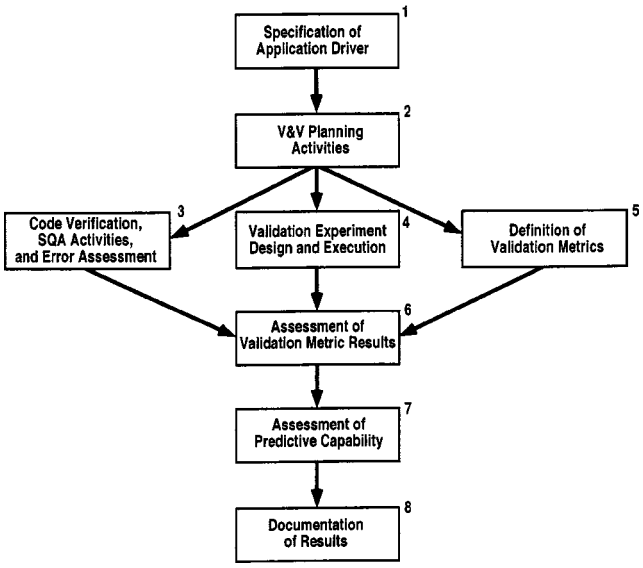


Fig. 6 Verification-validation-predictive process (Adapted from [30])

from a variety of sources, such as limited or nonexistent experimental data for a fixed (but unknown) physical parameter, limited understanding of complex physical processes, or insufficient knowledge concerning initial conditions and boundary condition in an experiment. Epistemic uncertainty is also referred to in the literature as reducible uncertainty, subjective uncertainty, and model form uncertainty.

Issues of acknowledged and unacknowledged errors are primarily discussed with regard to verification activities (Section 2.3). Issues of aleatory and epistemic uncertainty are primarily discussed with regard to validation and prediction capability activities (Sections 2.4 and 3.3).

Referring back to Fig. 5*c*, the requirement for use of a model ''far from'' the validation database necessitates extrapolation well beyond the physical understanding gained strictly from experimental validation data. As a result, the type of uncertainty that dominates our inference to the application domain is epistemic uncertainty. For example, the modeling of specific types of interactions or coupling of physical processes may not have been validated together in the given validation database. Examples of this situation are safety assessment of nuclear-reactors during failure conditions [49–52] and the assessment of nuclear-waste-repository performance [53–56]. There is then significant epistemic uncertainty in the accuracy of model predictions describing such interactions. Aleatory uncertainty will also be a factor in the inference, but the dominant contributor is lack of knowledge of the physical processes active in the extrapolation.

2.2 Modeling and simulation requirements

To improve the efficiency and confidence-building impact of V&V, we believe it is necessary to improve the coupling of V&V activities with the requirements of the intended application of the computational physics code. Figure 6 depicts a high-level view of the verification-validation-prediction process that begins with the specification of the application driver and the requirements for the computational physics code [30]. In Fig. 6 we suggest that the complete process relies upon the following activities:

1) Identification and specification of the application driver that focuses the use of the code under discussion
2) Careful planning of V&V activities, especially the use of the Phenomena Identification and Ranking Table (PIRT) to prioritize V&V activities for the application driver and preliminary specification of prediction accuracy requirements
3) Development, implementation, and documentation of code verification and SQA activities for the code, as well as solution error assessment of validation calculations
4) Design and execution of validation experiment activities in accordance with the PIRT
5) Development and definition of appropriate metrics for comparing computational results with experimental results to measure confidence in the intended application of the code
6) Assessing the results of the validation metrics with regard to preliminary specification of prediction accuracy requirements
7) Assessment of the predictive accuracy of the code using the validation metrics and the refined application requirements, with emphasis on high-level system response measures
8) Accurate and full documentation of the planning, results, and consequences of the validation activities, especially their implications for predictive confidence for the application driver of the code

Note that the although the activities shown in Fig. 6 are shown sequentially, the process is typically an iterative one.

This paper briefly discusses each of these eight activities, but [30] discusses each in detail. In this section we only comment on the two key activities that directly address the focus of V&V on the requirements of the application: block 1 ''Intended Application'' and block 2 ''Planning.'' The intended application is the application for which the modeling and simulation capability is being developed. Concerning application requirements, validation activities must assess confidence in the use of the code for a specified application. The application requirement at which a particular validation activity is directed is a critical planning element and must be defined before the performance of any specific validation work. One of the methods in risk assessment for identifying specific applications deals with identifying system-application scenarios. In the nuclear weapons area, event scenarios are segregated into three wide categories to which the engineering system might be subjected: normal, abnormal, and hostile environments. Although these categories may not be appropriate for all engineering systems, we believe they are a helpful framework for many types of systems, eg, systems ranging from military and commercial aircraft to spacecraft to power generation facilities and public buildings.

Regarding planning, we believe the PIRT is the most important tool in the planning process for translating technical requirements for the relevant code application into priori-

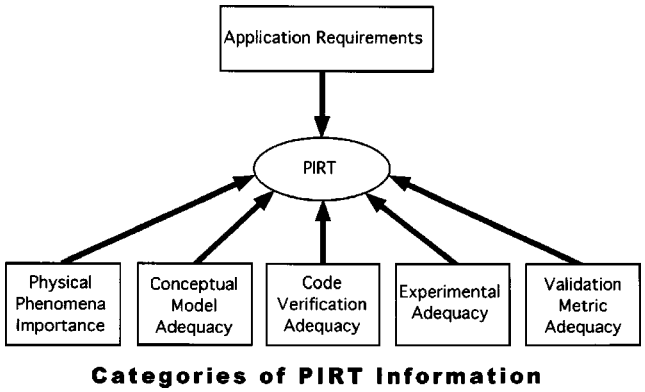


Fig. 7 Aspects of PIRT related to application requirements (adapted from [30])

tized requirements for V&V activities [29]. The PIRT was developed to assess the safety of nuclear reactors and has primarily been used for that purpose [57–61]. The PIRT is critical for planning validation experiments because it helps establish both sufficiency and efficiency of the validation activities. To demonstrate sufficiency requires a careful response to the question: What has to be done to establish a necessary level of confidence in the application of the code? To demonstrate efficiency requires evidence that limited resources (people, time, money) are balanced as a result of planning, not simply as a reaction to circumstances. We presume in this paper that dedicated validation experiments supporting the validation assessment of a particular code application are directed at the most important elements in the associated PIRT. If this is not true, there is already a revealed weakness in the planned validation activities. The planning for the dedicated validation experiments should make this direction explicit and clear.

The PIRT addresses five aspects of V&V that relate to the application requirements. Depicted in Fig. 7, these aspects are as follows:

1) Physical Phenomena Importance. The PIRT systematically identifies the physical phenomena that are needed for the modeling and simulation application requirements. Each phenomenon must be prioritized, and the criteria applied to accomplish this prioritization must be described. Occasionally, these criteria must span the scenarios of normal, abnormal, and hostile environments.
2) Conceptual Model Adequacy. The PIRT identifies the adequacy of the existing set of needed physical phenomena for the application requirements.
3) Code Verification Adequacy. The PIRT identifies the adequacy of the set of code-verification test cases.
4) Experimental Adequacy. The PIRT identifies the adequacy of existing experimental data required for model validation. If existing data are inadequate for the validation requirements, resources must be allocated and planning begun for the needed additional validation experiment tasks.
5) Validation Metric Adequacy. The PIRT identifies the validation metrics needed for the requirements and the preliminary accuracy requirements needed for these metrics. Two types of validation adequacy must therefore be addressed: 1) identification and specification of the validation metrics and 2) preliminary specification of the magnitude of the metrics needed to satisfy the application requirements. Stated differently, validation metrics appropriate for the application driver should be defined and estimates of the required predictive accuracy should be made. However, as stated previously, the requirement for the adequacy of all of the specified metrics is coupled to the overall application requirements; hence, the process of specifying the validation metrics will interact with the process of specifying the application requirements.

As stressed by Boyack [57], the PIRT is certainly not set in stone once it is formulated and documented. While a given formulation of the PIRT guides V&V activities, the PIRT must also be adaptable to reflect information gathered during the conduct of these activities. Importantly, to gain the greatest advantage of its value during planning, the PIRT can and probably should be adapted as experimental validation activities are conducted. Several different outcomes illustrate how experimental validation activities can interact with and influence changes to the PIRT:

- A validation experiment may be planned and conducted under the assumption that a specific PIRT element has high importance. After the results of the experiment are analyzed, the importance of that PIRT element is found to change from high to medium or low in response to these results. This does not argue that the underlying application requirements could or would change as a result of experiments, only that the technical importance of an element for validation may change.
- An experiment is conducted that reveals a shift of a PIRT element from low to high importance. This may require, for example, that a subsequent exploratory experiment be performed that was not identified in the existing PIRT.
- An experiment addressing a high-importance PIRT element is performed. The current code implementation addressing that phenomenon is believed to be adequate. However, it is discovered unexpectedly that the code cannot even function properly in defining the proposed experiments, thereby changing the ranking of the implementation to inadequate.
- An experiment designed to probe fully coupled phenomena reveals the presence of a completely unexpected and unknown phenomenon that is of high importance for the application driver. Not only must the PIRT be changed to reflect this event, but also the overall V&V effort for the code application may require significant revision. For example, a previously low-ranked phenomenon may now be ranked high or a planned validation experiment may have to be redefined as a phenomenon-exploration experiment.
- A validation experiment for a single phenomenon reveals that certain models implemented in the code must be recalibrated. This changes the code implementation from adequate to incomplete and may require additional planning for calibration experiments to improve the precalibration model capabilities.

2.3 Verification activities

2.3.1 Fundamentals of verification

Two types of verification are generally recognized in computational modeling: code verification and solution verification [14,16]. Because of recent work by several investigators [11,12], we now believe that code verification should be segregated into two parts: numerical algorithm verification and SQA. Numerical algorithm verification addresses the software reliability of the implementation of all of the numerical algorithms that affect the numerical accuracy and efficiency of the code. In other words, this verification process focuses on how correctly the numerical algorithms are programmed (implemented) in the code and the accuracy and reliability of the numerical algorithms themselves. This issue is of paramount importance in computational physics codes, whereas in conventional areas of application of SQA, such as real-time control systems, this issue receives less emphasis. The major goal of numerical algorithm verification is to accumulate sufficient evidence to demonstrate that the numerical algorithms in the code are implemented correctly and functioning as intended. SQA emphasizes determining whether or not the code, as a software system, is reliable (implemented correctly) and produces repeatable results on specified computer hardware and a specified system with a specified software environment, including compilers, libraries, etc. SQA focuses on the code as a software product that is sufficiently reliable and robust from the perspective of computer science and software engineering. SQA procedures are needed during software development and modification, as well as during production-computing operations.

Solution verification deals with the quantitative estimation of the numerical accuracy of a given solution to the PDEs. Because, in our opinion, the primary emphasis in solution verification is significantly different from that in numerical algorithm verification and SQA, we believe solution verification should be referred to as *numerical error estimation*. That is, the primary goal is attempting to estimate the numerical accuracy of a given solution, typically for a nonlinear PDE with singularities and discontinuities. Assessment of numerical accuracy is the key issue in computations used for validation activities, as well as in use of the code for the intended application.

The study of numerical algorithm verification and numerical error estimation is fundamentally empirical, ie, the study is based on observations and testing of the code for specific input options chosen. Numerical algorithm verification deals with careful investigations of topics such as spatial and temporal convergence rates, iterative convergence, independence of solutions to coordinate transformations, and symmetry tests related to various types of boundary conditions. Analytical or formal error analysis is inadequate in numerical algorithm verification: the code must *demonstrate* the analytical and formal results of the numerical analysis. Numerical algorithm verification is conducted by comparing computational solutions with highly accurate solutions. We believe Roache's description of this, ''error evaluation,'' clearly distinguishes it from numerical error estimation [62]. Numerical error estimation deals with approximating the numerical error for particular applications of the code when the correct solution is not known. In this sense, numerical error estimation is similar to validation assessment.

In our view, to rigorously verify a code requires rigorous proof that the computational implementation accurately represents the conceptual model and its solution. This, in turn, requires proof that the algorithms implemented in the code correctly approximate the underlying PDEs, along with the stated initial conditions and boundary conditions. In addition, it must also be proven that the algorithms converge to the correct solutions of these equations in all circumstances under which the code will be applied. It is unlikely that such proofs will ever exist for computational physics codes. The inability to provide proof of code verification is quite similar to the problems posed by validation. Verification, in an operational sense, then becomes the absence of proof that the code is incorrect. While it is possible to prove that a code is functioning incorrectly, it is effectively impossible to prove that a code is functioning correctly in a general sense. Single examples suffice to demonstrate incorrect functioning, which is also a reason why testing occupies such a large part of the validation assessment effort.

Defining verification as the absence of proof that the code is wrong is unappealing from several perspectives. For example, that state of affairs could result from complete inaction on the part of the code developers or their user community. An activist definition that still captures the philosophical gist of the above discussion is preferable and has been suggested by Kleindorfer *et al* [24] and stressed by Peercy [63]. In this definition, verification of a code is analogous to the development of evidence in a legal case. Thus, numerical algorithm verification and SQA activities consist of accumulating evidence substantiating that the code does not have any apparent algorithmic or programming errors and that the code functions properly on the chosen hardware and system software. This evidence needs to be documented, accessible, repeatable, and capable of being referenced. The accumulation of such evidence also serves to reduce the regimes of operation of the code where one might possibly find such errors.

The view of code verification in this paper as an ongoing process, analogous to accumulating evidence for a legal case, is not universally accepted. In an alternative view [14], code verification is not considered an ongoing process but one that reaches termination, analogous to proving a theorem. Obviously, the termination can only be applied to a fixed code. If the code is modified, it is a new code (even if the name of the code remains the same) and the new code must be verified again. In addition, all plausible nonindependent combinations of input options must be exercised so that every line of code is executed before one can claim that the entire code is verified; otherwise, the verification can be claimed only for the subset of options tested. The ongoing code usage by multiple users still is useful, in an evidentiary sense (and in user training), but is referred to as confirmation rather than code verification. In this alternative view of verification, it is argued that contractual and regulatory requirements for deliv-

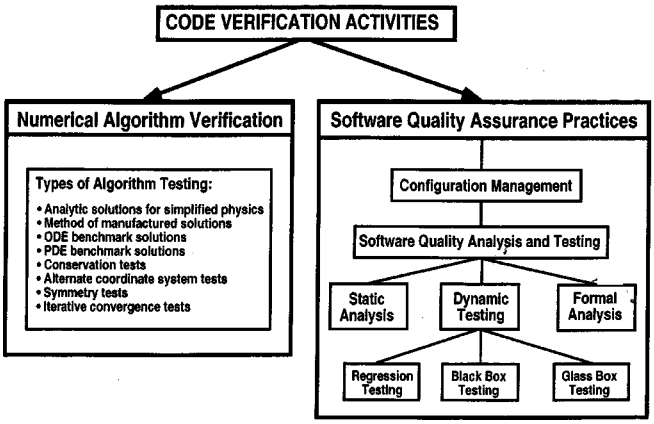


Fig. 8 Integrated view of code verification in computational physics (Adapted from [12])

ery or use of a ''verified code'' can more easily be met and that superficial practices are less likely to be claimed as partial verification. Ongoing usage of the code can possibly uncover mistakes missed in the code verification process, just as a theorem might turn out to have a faulty proof or to have been misinterpreted; however, in this view, code verification can be completed, at least in principle. Verification of individual calculations, as well as validation activities, are still viewed as ongoing processes.

Our view of integrating a set of code verification activities into a verification process for computational physics is conceptually summarized in Fig. 8. In this figure we have depicted a top-down process with two main branches: numerical algorithm verification and SQA practices. Numerical algorithm verification, which is the topic discussed in Section 2.3.2, focuses on the accumulation of evidence to demonstrate that the numerical algorithms in the code are implemented correctly and functioning properly. The main technique used in numerical algorithm verification is testing, which is alternately referred to in this paper as numerical algorithm testing or algorithm testing. The branch of SQA practices, discussed in Section 2.3.3, includes practices and procedures associated with SQA. SQA emphasizes programming correctness in the source program, system software, and compiler software. One of the key elements of SQA is configuration management of the software: configuration identification, configuration and change control, and configuration status accounting. As shown in Fig. 8, software quality analysis and testing can be divided into static analysis, dynamic testing, and formal analysis. Dynamic testing further divides into such elements of common practice as regression testing, black-box testing, and glass-box testing.

To the disbelief of many, a recent comprehensive analysis of the quality of scientific software by Hatton documented a dismal picture [64]. Hatton studied more than 100 scientific codes over a period of seven years using both static analysis and dynamic testing. The codes were submitted primarily by companies, but also by government agencies and universities from around the world. These codes covered 40 application areas, including graphics, nuclear engineering, mechanical engineering, chemical engineering, civil engineering, communications, databases, medical systems, and aerospace. Both safety-critical and non-safety-critical codes were comprehensively represented. All codes were ''mature'' in the sense that the codes were regularly used by their intended users, ie, the codes had been approved for production use. The total number of lines of code analyzed in Fortran 66 and 77 was 1.7 million and the total number of lines analyzed in C was 1.4 million. As the major conclusion in his study, Hatton stated, ''The T experiments suggest that the results of scientific calculations carried out by many software packages should be treated with the same measure of disbelief researchers have traditionally attached to the results of unconfirmed physical experiments.'' Hatton's conclusion is disappointing, but not at all surprising in our view. We also observe that both Stevenson [65] and Gustafson [66] strongly agree with Hatton's view that the problems uncovered by Hatton's static analysis experiments are basically independent of the programming language used.

We believe there are several reasons why disciplined testing of computational physics software is not a common practice. The first reason is that people working on computational physics codes do not like to be thought of as software engineers. A second reason is that the perceived risk associated with the incorrect functioning of nearly all computational physics codes is less than the perceived risk associated with the incorrect functioning of other kinds of software, such as safety-critical systems. For example, Johnson [67] discusses rigorous approaches to developing fault-minimal and fault-tolerant avionics software. There are regulatory requirements governing software verification for avionics applications [68] that are driven by the safety-critical consequences of software failure. We have never encountered a similar paper in computational physics that addresses the underlying consequences of software failure as Johnson addresses them.

The subject of software testing, and the potential risks and costs associated with inadequate testing, is the subject of a recent comprehensive study by the National Institute of Standards & Technology [69]. This report firmly states: ''Estimates of the economic costs of faulty software in the US range in the tens of billions of dollars per year...'' Precisely where risk-consequential computational physics software lies in this picture remains an important open question.

2.3.2 Numerical algorithm verification

Numerical algorithm testing focuses on numerical correctness and performance of the algorithms. The major components of this activity include the definition of appropriate test problems for evaluating solution accuracy and the determination of satisfactory performance of the algorithms on the test problems. Numerical algorithm verification rests upon comparing computational solutions to the ''correct answer,'' which is provided by highly accurate solutions for a set of well-chosen test problems. The correct answer can only be known in a relatively small number of isolated cases. These cases therefore assume a very important role in verification and should be carefully formalized in test plans for verification assessment of the code.

There are two pressing issues that need to be addressed in the design and execution of numerical algorithm testing. The first issue is to recognize that there is a hierarchy of confidence in highly accurate solutions. The AIAA Guide [9], for example, suggests the following hierarchical organization of confidence for the testing of computational physics codes: 1) exact analytical solutions, 2) semianalytic benchmark solutions (reduction to numerical integration of ordinary differential equations [ODEs], etc), and 3) highly accurate benchmark solutions to PDEs.

The second pressing issue in the design and execution of algorithm testing is to choose application-relevant test problems that will be used. There are two possible approaches for making this selection. One approach is to pick test problems that people have a great deal of experience with. It would be very advantageous if these problems developed into industry standards that could be broadly used in verification activities for specific engineering or physics disciplines. Unfortunately, no industry-standard test problems currently exist. A second approach is to construct specialized test problems that address specific needs that arise in the structure of the test plan. These test problems are specifically constructed to exercise the portions of the software that one requires.

Both existing test problems and specially constructed test problems can be either analytical solutions or benchmark solutions. Analytical solutions are closed-form solutions to special cases of the PDEs that are represented in the conceptual model. These closed-form solutions are commonly represented by infinite series, complex integrals, and asymptotic expansions. Numerical methods are usually used to compute the infinite series, complex integrals, and asymptotic expansions in order to obtain the solutions of interest. However, the accuracy of these solutions can be quantified much more rigorously than can the accuracy of the numerical solutions of the conceptual model. When computational solutions are compared with highly accurate solutions, either the comparisons should be examined along boundaries of interest or the error norms should be computed over the entire solution domain. The accuracy of each of the dependent variables or functionals of interest should be determined as part of the comparisons. Note that the most significant practical shortcoming of analytical solutions is that they exist only for very simplified physics and geometries.

A technique for developing a special type of analytical solution to be used for testing numerical algorithms is the ''Method of Manufactured Solutions'' (MMS) [14,70–73]. The MMS is a method of custom-designing verification test problems of wide applicability, where a specific form of the solution function is assumed to satisfy the PDE of interest. This function is inserted into the PDE, and all the derivatives are analytically derived. Typically these derivatives are derived by using symbolic manipulation software such as MACSYMA© or Mathematica©. The equation is rearranged such that all remaining terms in excess of the terms in the original PDE are grouped into a forcing-function or source term. This source term is then considered to be simply added to the original PDE so that the assumed solution function satisfies the new PDE exactly. For example, in the Navier-Stokes equations this term can be considered to be a source term, ie, a new term on the right-hand side of the PDE. The boundary conditions for the new PDE can be chosen to be the value of the solution function on the boundary (Dirichlet condition), a Neumann condition that can be analytically derived from the solution function, or a boundary condition of the third kind. This approach could be described as finding the problem, ie, the PDE, for which we have assumed a solution.

Using the MMS in code verification requires that the computed source term and boundary conditions are programmed into the code and that a numerical solution is computed. This technique verifies a large number of numerical aspects in the code, such as the numerical method, differencing technique, spatial-transformation technique for grid generation, grid-spacing technique, and correctness of algorithm coding. Although the MMS has been used in various forms for checking computer codes for 30–40 years, recent extensions and generalizations of the method have proven very effective. As pointed out by a number of researchers in this topic, solutions in MMS must be carefully chosen to achieve the desired results. For example, solutions should be chosen so that as many terms as possible in the original PDE are brough into play. This includes any submodels affecting terms in the original PDE, as well as any mathematical transformations of physical space to computational space.

Knupp and Salari [73] have systematically applied the MMS to Burger's equation, and the compressible and incompressible Navier-Stokes equations for laminar flow. These researchers have also presented an interesting study of the ''bug-finding'' capability of the method to better understand what errors the MMS is capable of resolving. In the study, the performance of the MMS on a set of 21 different coding mistakes was examined. The MMS correctly diagnosed every error in the set that prevented the governing equations from being solved correctly. Knupp and Salari (see also [14]) noted, however, that the MMS is incapable of detecting errors such as algorithm-efficiency mistakes, which occur in situations where the equations are still solved correctly by the algorithms but the coding is less efficient than is optimal. These results suggest the desirability of coupling studies like Hatton's T experiments with a dynamic testing diagnosis like the MMS to better correlate potential structural flaws with actual incorrect numerical performance. Such a study has yet to be performed.

In numerical algorithm testing the key feature to determine is the observed, or demonstrated, order of accuracy from particular numerical solutions. This should then be compared with the formal order of accuracy of the numerical method. Researchers have found a number of reasons why the observed order of accuracy can be less than the formal accuracy. Some of the reasons are: 1) a programming error exists in the numerical algorithm, 2) insufficient grid resolution so that the grid is not in the asymptotic convergence region of the Taylor series expansion for the particular solution variable of interest, 3) the formal accuracy for interior grid points is different from the formal accuracy for boundary conditions with derivatives resulting in a mixed order of

accuracy, 4) singularities, discontinuities, and contact surfaces interior to the domain of the PDE, 5) grid clustering, 6) singularities and discontinuities in the boundary conditions, 7) inadequate convergence of an iterative procedure in the numerical algorithm, and 8) over-specified boundary conditions. It is beyond the scope of this paper to discuss these in detail, however some of the representative references in these topics are [14,73–84]. It is our view that most of these reasons for degradation in formal order of accuracy are poorly understood and significant research is needed in this area.

Returning to the topic of types of highly accurate solutions, benchmark ODE solutions are very accurate numerical solutions to special cases of the general PDEs. These solutions commonly result from simplifying assumptions, such as simplified geometries that allow solutions in the formation of similarity variables. Benchmark PDE solutions are also very accurate numerical solutions to special cases of the PDEs or the boundary conditions. The accuracy of the benchmark solutions, whether ODE or PDE solutions, clearly becomes more of an issue as one moves away from analytical solutions. In the literature, for example, one can find descriptions of computational simulations that are considered to be of high accuracy by the author but are later found to be lacking. It is strongly recommended that no published solution be considered a numerical benchmark solution until: 1) the code used in producing the solution has be thoroughly verified and documented, 2) very comprehensive numerical error estimation is reported with the solution, and 3) the solution has been accurately calculated by independent investigators, preferably those who use different numerical approaches and computer codes.

Consistency tests can also be usefully employed as numerical algorithm tests. Global as well as regional tests can be made for the conservation of mass, momentum, and energy [85]. A number of tests can also be made that are related to the effect of numerical boundary conditions on the solution. One group of tests is used to determine if the same numerical solution is obtained in different coordinate systems [86]. Another group of tests is used to evaluate whether certain symmetry features are preserved in the solution. For example, if a plane of symmetry exists in the conceptual model, then the normal gradient of appropriate variables can be set to zero and a solution can be obtained. The same solution should also be obtained if this plane-of-symmetry condition is not imposed and the entire domain is solved. For unbounded domains, the boundaries of the computational domain are conceptually considered to be at infinity, ie, the boundaries are infinitely far from the spatial region of interest. Typically, a user-defined parameter specifies how ''far out'' these boundaries are. If the boundaries are too close, the asymptotic conditions applied there may not be accurate. The usual method of determining the size of the computational domain is to systematically increase the domain until the solution is no longer dependent on the size of the domain that is compatible with the objectives of the computation. Consistency tests should be considered as complementary to the other types of algorithm tests described in this section.

We believe that the primary responsibility for numerical algorithm verification should be placed upon the code development team. Computational analysts who are not directly a part of this team may conduct the algorithm testing, but we believe the code development team needs to have in-depth knowledge of the algorithms used and their expected performance. This information should be documented either as part of the code development process or during testing. The documentation on algorithm testing may be combined with that for SQA testing, as discussed in Section 2.3.3. It is important that analysts and code customers have access to this information.

2.3.3 Software quality assurance

Software quality analysis and testing, as highlighted in Fig. 8 presented previously, rests in three techniques: static analysis, dynamic testing, and formal analysis [87]. Static analysis techniques analyze the form, structure, and consistency of the code without executing the code. Examples of static analysis techniques are software reviews and inspections, complexity analyses, audits, and analyses of data flows. Dynamic testing techniques involve execution of the code. The results of executing the code are analyzed to detect coding errors or weaknesses in design that can cause coding errors. Regression testing, which reevaluates the accuracy of computed results, is one example of a dynamic testing technique. Formal analysis, also referred to as formal methods, is directed toward rigorously demonstrating that the code exactly represents the underlying conceptual model. Further remarks on static, dynamic, and formal analysis are included below. Two particularly useful books for readers who wish to acquaint themselves with the vast subject of SQA testing are by Beizer [88] and by Kaner *et al* [89]. The differing viewpoints in these books add interesting nuances to a subject as seemingly dry (to practitioners of computational physics) as SQA testing. Beizer's book is somewhat more formal in its approach, while Kaner and his colleagues are less structured in their approaches and recommendations.

It is probably safe to claim that static analysis has traditionally been dominated by techniques to detect compilation errors on the specific code of interest. One does not tend to see published studies of, for example, applying architectural analysis, such as the complexity metrics used in Hatton's analysis [64], to computational physics codes. We strongly advocate the expansion of static analysis as part of verification assessment for computational physics codes. The complexity of modern programming languages (C++ especially) and the use of massively parallel computing platforms for computational physics increase the need for, and the impact of, static analysis.

When considering software testing, it is important to understand the distinction between glass-box testing and black-box testing. Glass-box testing (also called structural testing) refers to testing when all of the source code of interest is available. Black-box testing (also called functional testing) refers to testing when only the code inputs and code outputs are observable. Glass-box testing is practiced primarily by code developers because it assumes sufficient knowledge of

the design and architecture of the code to design, populate, and assess test plans based on this knowledge. Glass-box testing is heavily weighted to what we have called SQA testing, although that is not mandatory. When we discuss static analysis, test problems generated by coverage analysis, and regression testing, we will be discussing examples of glass-box testing. We note that the MMS, discussed in Section 2.3.2, is actually code invasive and thus related to glass-box testing. The MMS cannot be applied exclusively in black-box mode because of the need for specific coding intervention for each test that it creates, at least in the examples of which we are aware. Once that coding is accomplished, however, the gathering of results proceeds in a fully black-box manner. There does not appear to be any analog of the MMS testing methodology in the work of Beizer [88] and Kaner *et al* [89], which is rather surprising.

Black-box testing is primarily the paradigm for the algorithm testing discussed above. Black-box testing can be performed by anyone involved with a given computational physics code, but it tends to be associated with independent testing entities. Certainly in the computational physics world, much black-box testing is performed by users of the code. (If a developer of the code is also a user, for purposes of this discussion we emphasize their role as a user.) Black-box testing requires no detailed knowledge of the code software, although such knowledge can certainly help. Even when those who designed the algorithms and implemented the code execute a test problem to investigate the empirical performance of the code, they are performing black-box testing. When a computational physics algorithm is described in a publication, for example, and compared with some highly accurate solution, a black-box test has been performed. The goal of the authors in that case is to assess the functionality and accuracy of the output results, not to test specific software elements. When users test codes by running their favorite test problems prior to application, they are performing black-box testing. Users are an enormous resource for black-box testing. In our discussion of dynamic testing, aspects of regression testing, and statistical testing, we are addressing issues that are particularly relevant to black-box testing. The goal of a coordinated, optimal test strategy for a computational physics code is to blend black-box testing and glass-box testing.

Coverage analysis and code sensitivity analysis are methods that can be used to assess the complexity of the computational physics code in order to design collections of test problems. Coverage analysis enables one to determine what components of the code enter into various options exercised in the code. The relative importance of the selected component contributions to that calculation is then determined via a sensitivity analysis. Ideally, one would start coverage analysis with a tool of sufficient capability to assess code lines and units exercised by the code in performing the desired calculation. For example, PureCoverage™ [90] is a commercial tool that detects lines of code executed during the operation of C++ software and reports this information in a useful form. A major challenge of coverage analysis for algorithm testing is to understand execution paths as well as lines touched in the code in order to execute the intended application calculation. One then designs tests that specifically target these modules and paths. This is clearly a glass-box-testing strategy and is discussed extensively in both Beizer [88] and Kaner *et al* [89]. While a discussion of sensitivity analysis is beyond the scope of this paper, the topic is brought up in the context of software quality testing in [91].

A natural partner of coverage analysis for software quality testing is regression testing. A widely used commercial testing technique, regression testing is defined by Beizer [88] as ''any repetition of tests (usually after software or data change) intended to show that the software's behavior is unchanged except insofar as required by the change to the software or data.'' What this means in practice for computational physics is that a compendium of tests is assembled, a baseline for code performance against these tests is established, and the tests are run periodically. When the current performance is determined to have deviated from the baseline, either the decision is made that a bug has been introduced or the regression test suite is baselined again. There are two requirements that make this approach feasible. First and foremost, the test suite must run sufficiently fast so that the testing can be performed frequently. Second, the regression testing should be performed for every software modification. For multiple developers of a computational physics code, for example, this means that no modified software is accepted before regression testing is performed. This, of course, can impede the software development process if not managed properly.

Regression tests that are glass-box tests are often correlated with coverage analysis and designed to provide testing of as many lines of code as possible. Systematic, cyclic-regression testing, once a suitable set of test problems has been defined, is known to be important in code development and maintenance. Unfortunately, this requires a trade-off between available resources that influences the fidelity of the testing. For example, a high-fidelity test suite that takes longer than 5 to 10 hours to execute is probably not practical for execution every day. Thus, such a suite cannot be used to demonstrate the stability of daily code builds.

The main use of regression testing is to minimize the effort devoted to fixing incorrect new software that is introduced into existing code in a software development project. There is a balance that must be determined between the effort devoted to regression testing and the effort devoted to finding and fixing incorrect modifications of the existing code at a later date. At the same time, regression testing aims to cover as much of the code as possible. In the computational physics projects that we are familiar with, the anticipated coverage target for regression testing is typically on the order of 80% of the lines of code. Accepting 80% line coverage is commonly based on the developer's willingness to define specific code elements as not requiring coverage. More importantly, these coverage thresholds are established in the attempt to prevent the effort devoted to regression testing from becoming larger than the effort required to fix introduced errors.

A great deal of effort has recently been devoted to the

development and application of formal methods [92–94]. However, the actual goal of these methods—rigorous ''proof'' that a system of software is correctly implemented—remains controversial in our opinion. The application of formal methods is also complicated by disputes over cost, appropriateness, utility, and impact [95]. Formal methods have certainly not been applied to software systems like those of interest in computational physics, namely, those systems in which floating-point arithmetic is dominant and an effectively infinite variability in software applications is the norm. The utility of these methods to computational physics codes, even if resource constraints were not issues, has not been established. This fact has led to interest in the application of formal methods to more restricted problems that still may be of significant interest in computational physics software. For example, there is current interest in applying formal methods to aid in verification of the mathematical formalism of the conceptual model that underlies a computational physics code rather than in the full details of the software implementation [96,97].

We conclude this section on SQA by providing a few brief remarks and references for some of the key literature that is relevant to the overall problem of verification assessment. A number of modern texts are available that describe current practice in the SQA field [88,98–107]. Much of the research and development in SQA procedures has been fueled by computer-controlled systems that require extremely reliable and secure software, as in Johnson's avionic-systems application referenced previously. Such systems are commonly referred to as ''high-integrity systems.'' Examples of other high-integrity systems are control systems for nuclear power reactors and software for nuclear-weapon security and safety. The scientific software community has much to learn from SQA procedures that have been developed for these systems as high-consequence scientific computing becomes more prevalent. This includes an improved understanding of the responsibilities associated with producing high-integrity computational physics codes.

2.3.4 Numerical error estimation

The two basic approaches for estimating the error in a numerical solution to a PDE are *a priori* and *a posteriori* approaches. An *a priori* approach uses only information about the numerical algorithm that approximates the partial differential operators and the given initial and boundary conditions. *A priori* error estimation is a significant element of classical numerical analysis for PDEs, especially those underlying the finite element and finite volume methods [14,78,108–112]. An *a posteriori* approach uses all of the *a priori* information, plus computational results from a previous numerical solution using the same numerical algorithm on the same PDE and initial and boundary data. In this paper we also refer to *a posteriori* error estimates as ''empirical'' data or, as Roache [14] calls it, ''observed'' data. Empirical data could be, for example, a solution on a single coarse grid or a sequence of solutions on consecutively finer grids. We believe the only quantitative assessment of numerical error that can be achieved in practical cases of nonlinear PDEs is through *a posteriori* error estimates. As a result, only *a posteriori* will be discussed here.

A posteriori error estimation has primarily been approached through the use of either Richardson extrapolation [14] or estimation techniques based on finite element approximations [113,114]. Richardson's method can be applied to any discretization procedure for differential or integral equations, eg, finite difference methods, finite element methods, finite volume methods, spectral methods, and boundary element methods. As pointed out by Roache [14], Richardson's method produces different estimates of error and uses different norms from the traditional *a posteriori* error methods used in finite elements [109,115]. A Grid Convergence Index (GCI) based on Richardson's extrapolation has been developed to assist in the estimation of grid convergence error [14,116,117]. The GCI converts error estimates that are obtained from any grid-refinement ratio into an equivalent grid-doubling estimate. Recent studies have shown that the GCI method is fairly reliable, even for solutions that are not in the asymptotic-convergence region [118-121].

Richardson's extrapolation can be applied to compute error estimates of dependent variables at all grid points, as well as error estimates for solution functionals. Solution functionals are integrated and differentiated quantities such as body lift and surface heat flux, respectively. Venditti and Darmofal [122] discuss the coupling of error estimation and adaptive grid generation to reduce numerical errors in such computed functionals. Their particular example is the quasi-1D flow in a variable-area duct. It is important to emphasize that different dependent variables and functionals converge at different rates as a function of grid size. For example, the grid and time step that are required to show second-order convergence in heat flux on a body surface are extremely finer than for total lift on a body. This fact was also pointed out by earlier researchers [14,123], as well as in the AIAA Guide [9].

A posteriori error estimates are also important for finite element adaptivity, where both the spatial grid density (h-adaptivity) and the order of the finite element scheme (p-adaptivity) can be adapted. The important role of *a posteriori* estimates in adaptivity for finite elements is discussed in many papers, as well as in the recent books by Ainsworth and Oden [113] and Babuska and Strouboulis [114]. For numerical error estimation, however, finite element adaptivity is not the goal. *A posteriori* methods used in finite elements are very computationally efficient and recently these methods have been developed for practical engineering error measures; both global and local measures. Also, recent work has addressed error estimation for certain types of singularities and nonlinearities in elliptic problems [113,114], and nonlinear parabolic problems [124,125]. Regardless of the *a posteriori* error estimation approach chosen, the estimation should always be made for the system response outputs that are relevant to the application of interest.

The assumption of smoothness in solutions, ie, no singularities and discontinuities, is quite demanding in estimating local errors in the solution domain. Singularities and discontinuities commonly occur in fluid dynamics, solid mechanics, and structural dynamics. The ''pollution'' of particular regions of a calculation by the presence of singularities such as shock waves, geometrical singularities, or crack propagation

is a subject of grave concern in error estimation. Often, the only clear sense of this pollution available to us is through careful empirical assessment. Since the technical aspects of this issue are beyond the scope of this paper, the reader should consult a series of papers by Babuska and his colleagues [126–128] as well as the paper by Oden [129] for a discussion of this problem from a finite-element point of view. Roache [14] has a wide-ranging discussion of this topic, and the recent work of Zhang and his colleagues [130] discusses how the presence of a shock wave structure affected *a posteriori* error estimates for the Euler equations. A recent paper of Botella and Peyret [82] discusses similar problems associated with computing singular solutions of the Navier-Stokes equations.

An additional numerical error estimation complexity should be mentioned that can occur in certain types of computational physics simulations. This complexity can be conceptually described as a coupling between numerical error and the appearance of new spatial and temporal scales in certain types of physical models. In fluid dynamics, for example, when grids are sufficiently refined, completely new processes or characteristics on the smaller spatial scales can develop. It is not uncommon that a steady flow field can become unsteady as the grid is refined and very small-scale phenomena can develop that did not exist on the coarser grids. This occurs in the case of the trailing edge region of a compressor or turbine blade, where vortex shedding is known to exist and will appear in the simulation if the grid around this region is sufficiently refined. Refining grids does not ensure that the physics modeled will remain unchanged as the grid is resolved. This observation directly influences the accuracy and reliability of any type of *a posteriori* error estimation method, especially extrapolation methods.

Extracting an estimate of numerical accuracy on underresolved grids is a topic of current research. We previously mentioned one aspect of the problem, generically referred to as pollution error—the propagation of discretization error from less accurately resolved regions of a calculation into more accurately resolved regions. Another issue is simply to understand what information can rationally be synthesized from calculations with more- or less-known resolution accuracy. Though this topic is beyond the scope of this paper, Chorin [131–133] and Glimm [134,135] demonstrate that techniques for understanding the numerical accuracy of underresolved calculations are challenging and specialized. A critical theme in this literature, and one that might be generalized to other important problems, is the role of probability and statistical estimation in performing such assessments. We conclude by commenting that studying V&V for underresolved calculations directly addresses the challenges posed by Gustafson in his 1998 paper [66].

2.4 Validation activities

2.4.1 Fundamentals of validation

We have emphasized that validation experiments must be designed purposefully with specific goals linked to application objectives and to specific elements of the PIRT. The primary goal of directed validation experiments is to ensure that experimental data are sufficiently related to the application driver to provide stringent confidence assessment when code calculations are compared with these data. These data must be precisely located within the structure of the application-specific PIRT and must be unambiguous in defining confidence in the application of the code to the phenomena exemplified in the experiment. Because this task must be accomplished by quantitatively comparing code calculations with experimental data, important requirements are placed on validation experiments to create the greatest opportunities for performing these comparisons. It is critically important to design and execute validation experiments that allow precise and conclusive comparisons of calculations with experimental data for the purpose of assessing model fidelity and credibility. In Section 2.4.5 we further discuss model-experiment comparisons. It will hopefully be made clear that the design and execution of experiments must allow us to quantify meaningful and useful metrics.

Particular validation experiments may achieve these goals to a greater or lesser degree, but any such attempt rests on a foundation of rational design concepts that are based on the needs of the application of the code. The more purposefully we design an experimental validation activity, the greater the probability that we can optimize the results of that activity in the face of these complex constraints. It is important from an engineering perspective that validation experiments balance resource constraints, including time, level of effort, available expertise, and desired fidelity. The approach for achieving this balance should be defined in the experimental plan. How to achieve this balance will depend strongly on which tier of physics complexity the experiment is conducted. As briefly mentioned in Section 1.2, validation experiments can be conducted at the unit-problem, benchmark, subsystem, or system tier. The resources needed vary greatly from one tier to the next. Section 2.4.2 discusses some of these topics in more detail.

The experimental, computational, and comparison activities should expedite the quantitative assessment of computational models for system applications that are within the application domain. It is important to attempt to quantify the boundary separating the region of acceptability of the model from the region where the model is not acceptable for the application. Designing experiments that test a model in regions where the model is believed to be insufficiently accurate for the intended application helps locate this boundary and provides a means for quantitatively assessing the degree of the expected inaccuracy. Because such experiments are performed purposefully rather than accidentally, these experiments also further test our grasp of the conceptual models underlying the code that are probed by the validation experiments. Obviously, this goal only makes sense when experiments that probe model inaccuracy lie close enough to the boundary of applicability to be relevant. It is desirable to have experimental validation tasks that have the explicit goal of defining those application domains where use of the model is questionably adequate to better quantify the boundary of applicability of the model. To perform these validation

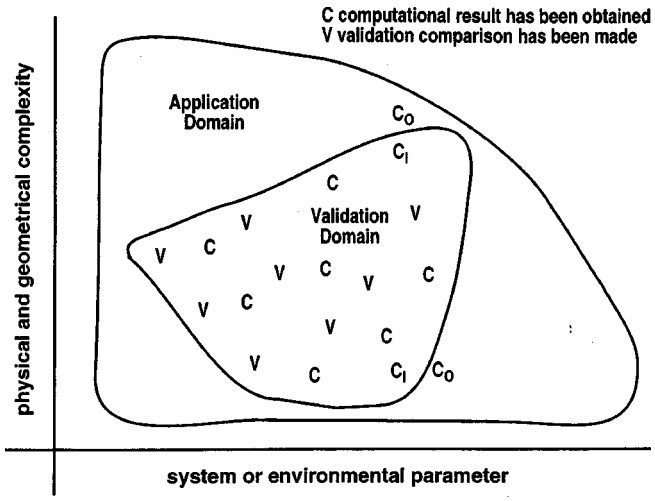


Fig. 9 Typical relationship between the application domain and the validation domain (adapted from [30])

tasks in a conscious and scientifically directed manner is not necessarily any easier than to achieve desirable levels of confidence in other application domains.

Figure 9 illustrates a typical relationship between the application domain and the validation domain. The relationship between the application domain and the validation domain shown in Fig. 9 is actually part of the class of relationships shown in Fig. 5*b*: the application domain is not a subset of the validation domain. Two kinds of points are shown in Fig. 9. A ''C'' denotes a location in the parameter-complexity space where a computational result in the application domain has previously been performed; a ''V'' denotes a location in the space where a validation comparison between computation and experiment has been performed. The boundary of the validation domain represents the apparent limit where the physical models in the code would no longer be reliable. The boundary of the validation domain could be estimated, for example, by using alternate plausible computational models or by using expert opinion. We draw the reader's attention to two pairs of intended applications, each pair denoted ''C_I'' and ''C_O''. The applications ''C_I'' each lie inside of the validation domain, while the applications ''C_O'' each lie outside of the apparent validation domain. Our point is that it is essentially as important from an engineering project perspective to know that the model is questionable for the applications ''C_O'' as it is to know that the model is acceptable for the applications ''C_I'' [136,137]. An extensive validation effort could identify the validation boundary accurately, but such an effort is commonly not possible, for example, because of programmatic schedules, budgets, safety and environmental concerns, or international treaties.

There should be a well-reasoned basis for choosing between two potential validation experiments. The basis should depend on the potential benefits of defining the acceptable domain boundary and on the use of available time and resources. The prioritizations embodied in the PIRT should be helpful in choosing between the two potential experiments. In partnership with the PIRT, two approaches are recommended: competing alternate plausible models and statistical experimental designs. Competing alternate plausible models can be of two different types: alternate models that are generally at the same level of modeling fidelity, and hierarchical models. Using hierarchical models, one formulates conceptual models that are clearly at different levels of fidelity or physical detail. Results from the models at different levels of complexity are compared to attempt to assess where the lower fidelity model could begin to fail. Statistical experimental design (for example, see Cox [138] and Dean and Voss [139]) is an attractive basis for attacking the resource optimization problem for experiments with a focus on uncertainty quantification. Gunter [140] has argued in favor of using statistical experimental design in planning physical-science experiments. The specifics of how the statistical design of experiments may be applied in particular validation activities are dependent upon the subject-matter focus of these experiments.

2.4.2 Construction of validation experiment hierarchy

Any appropriate methodology for the purposeful definition, design, and analysis of experiments for validation of a code must consider the intended application. As discussed in Section 2.2, the PIRT is the key methodology for identifying the predictive-capability needs and setting the validation priorities. To implement the needs and priorities identified in the PIRT, the recommended methodology is the use of a validation hierarchy, as discussed in Section 1.2. The hierarchical view of validation is distinctly an engineering perspective, as opposed to a scientific or model-building perspective. The purpose of the validation hierarchy is to help identify a range of experiments, possible separation of coupled physics, and levels of complexity—all of which are related to an engineering system—so that computer codes from different disciplines can be evaluated. Stated differently, the validation hierarchy must be *application* driven to be of engineering value, *not code capability* driven.

As one constructs each lower tier of the hierarchy, the emphasis moves from multiple coupled-physics codes to single codes (or single-physics options in multi-physics codes) simulating a particular type of physics. In parallel with the simplification process, the focus on the actual operating conditions of the complete system should not be lost. Constructing these hierarchical tiers and identifying the types of experiments that should be conducted at each tier are formidable challenges. There are many ways of constructing the tiers; no single construction is best for all cases. We would draw the analogy of constructing validation hierarchies to the construction of control volumes in fluid-dynamic analyses. Many varieties of control volumes can be drawn; some lead nowhere, and some are very useful for the task at hand. The construction should emphasize the modeling and simulation capability that is desired to be validated, whether that capability is computational physics or other computational disciplines. Analogous tier structures can be developed for structural dynamics and electrodynamics, for example, when the engineering system of interest involves these disciplines.

A good hierarchical tier construction is one that accom-

plishes two tasks. First, the construction carefully disassembles the complete system into tiers in which each lower-level tier has one less level of physical complexity. For complex engineered systems, this may require more than the three building-block tiers shown in Fig. 2, presented previously. The types of physical complexity that could be uncoupled from one tier to the next are spatial dimensionality, temporal nature, geometric complexity, and physical-process coupling. The most important type of physical complexity is physical-process coupling, which commonly contains the highest nonlinearity of the various contributors. The nonlinear coupling is, of course, eliminated by conducting experiments with uncoupled and separate physical effects, eg, separating turbulence and combustion. It is important to recognize the potential nonlinear nature of all of the contributors in the construction of the tiers because the philosophy of the tier construction rests heavily on linear-system thinking. That is, the belief that confidence in the computational capability for the complete system can be built from an assessment of the computational capability of each of its parts. The complete systems of interest clearly do not have to be linear, but the philosophy of the hierarchical validation approach loses some of its utility and strength for strong nonlinear coupling from one tier to the next.

The second task accomplished by a good hierarchical tier construction is the selection of individual experiments in a tier that are practicably attainable and able to produce validation-quality data. In other words, the individual experiments should be physically achievable given the experimental test facilities, budget, and schedule, and they should be capable of producing quantitative experimental measurements of multiple system-response measures that can test the model in question. For complex systems, the ability to conduct a true validation experiment at the complete system tier is extremely difficult, if not impossible. At the subsystem tier, it is usually feasible to conduct validation experiments, but it is still quite difficult and expensive. One usually chooses a single hardware subsystem or group of subsystems that are closely related in terms of physical processes or functionality. For complex subsystems, one might want to add a new tier called ''components'' below the subsystem tier. As with the subsystem tier, the components tier would consist of actual operational hardware components. When one defines the individual experiments at the benchmark-tier level, special hardware should be fabricated. By ''special hardware'' we mean that the hardware need not be constructed to perform its system tier or subsystem tier functions.

The benchmark tier is probably the most difficult to construct because it represents the transition from a hardware focus in the two top tiers to a physics-based focus in the bottom tiers of the hierarchy. At the bottom tier, unit problems, one should identify simple geometry experiments that have a single element of physical-process complexity. As with the subsystem tier, an additional tier may need to be added to attain only one element of physics at the bottom tier. Also, the experiment must be highly quantitatively characterized so that it can provide the necessary data to the computational code, and the experiment must be conducted so that experimental uncertainty can be estimated precisely. High-quality validation experiments are practicably attainable at the benchmark and unit-problem tiers, but usually not at the system or subsystem tiers for complex systems.

Recently, an example of a hierarchical tier structure for a complex, multidisciplinary system was presented in [11]. The example features an air-launched, air-breathing, hypersonic cruise missile. The missile is referred to as the complete system, and the following are referred to as systems: propulsion, airframe, guidance, navigation, and control (GNC), and warhead. The example assumes that the missile has an autonomous GNC system, an on-board optical target seeker, and a warhead. Figure 10 shows the system-level hierarchical validation structure for the hypersonic cruise missile. The structure shown is not unique, nor is it necessarily optimum for every computational-simulation perspective of the missile system. In addition, the structure shown in Fig. 10 focuses on the airframe system and the aero/thermal protection subsystem for the purpose of analyzing the aero/thermal performance of the missile.

To better demonstrate how the validation hierarchy of the aero/thermal protection subsystem is related to the validation hierarchy of the propulsion, airframe, GNC, and warhead systems, the example was discussed further in [11]. Figure 11 shows how the validation hierarchy of each of these four systems could be viewed as the primary facets of a four-sided pyramid. The airframe facet was divided into three additional facets, each representing the three subsystems: aero/thermal protection, structural, and electrodynamics. The propulsion system was divided into four additional facets to represent its subsystems: compressor, combustor, turbine, and thermal signature. Similarly, the GNC and the warhead systems could be divided into subsystems appropriate to each. On the surface of this multifaceted pyramid, one can more clearly and easily indicate the coupling from one facet to another. For example, we discussed the coupling of laminar and hypersonic flow with ablation to the optical seeker of the GNC system. This coupling would be shown by an arrow connecting these hypersonic flow elements to appropriate elements on the GNC facet of the pyramid. (Suggested references stressing a systems engineering perspective are [27,34,141–143].)

The validation pyramid stresses the systems-engineering viewpoint in modeling and simulation-based design, as opposed to the viewpoint of a specific discipline. Each facet of the pyramid can then be devoted to identifying validation experiments for each computational model responsible for part of the design of the complete system. As one traverses around the top of the pyramid, the number of facets is equal to the number of systems that are identified. As one traverses around the bottom of the pyramid, the number of facets is equal to the total number of major computer codes used in the analysis of the engineering system, ie, the number of codes that require validation activities for the intended application. For the example of the hypersonic cruise missile, if the code that simulates surface ablation is a separate code from the aerodynamics code, then an additional facet on the

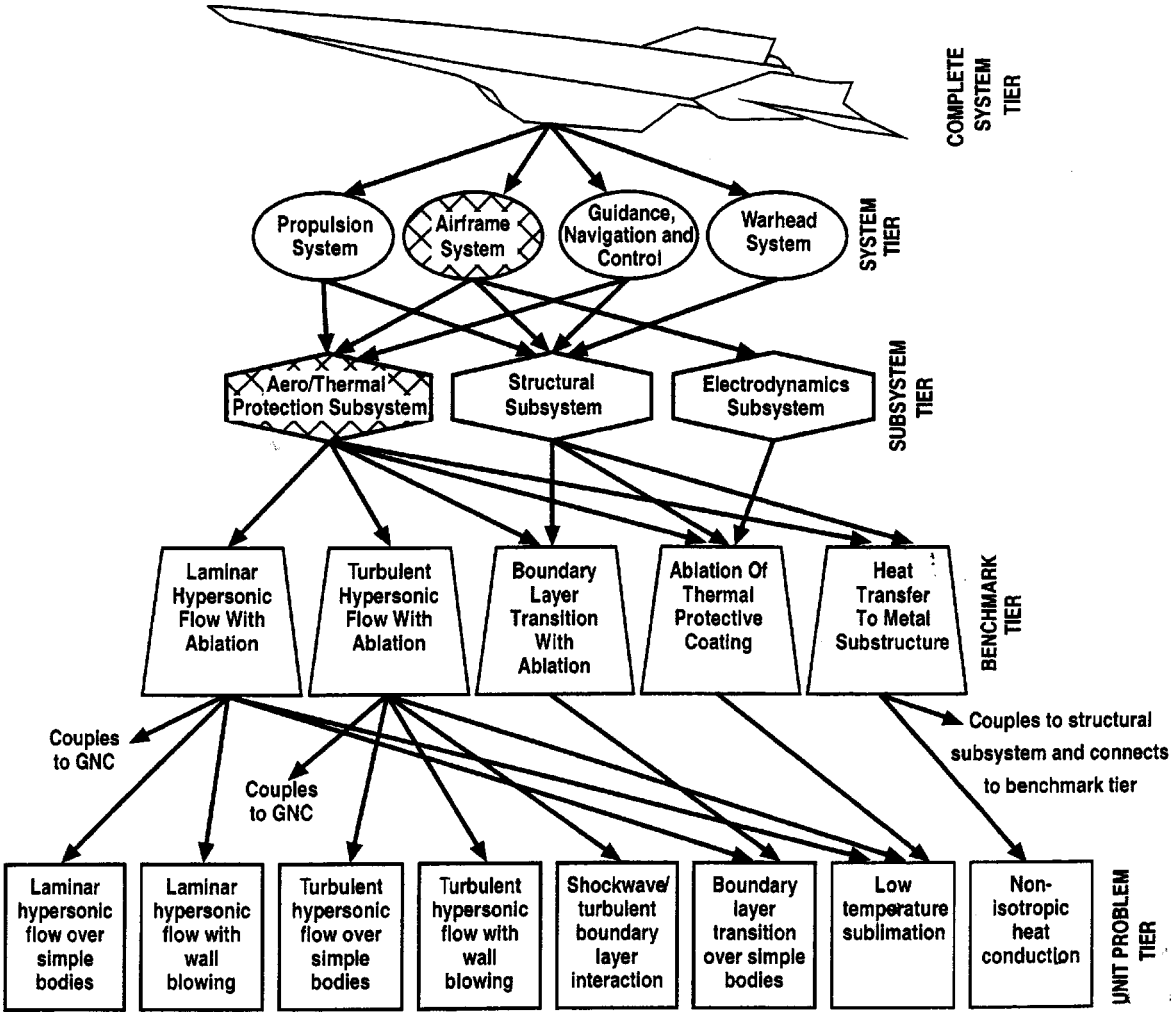


Fig. 10 Validation hierarchy for a hypersonic cruise missile [11]

pyramid is added on the aero/thermal subsystem facet. We strongly believe this type of system-level thinking is necessary to increase confidence in complex systems that are designed, manufactured, and deployed with reduced levels of testing.

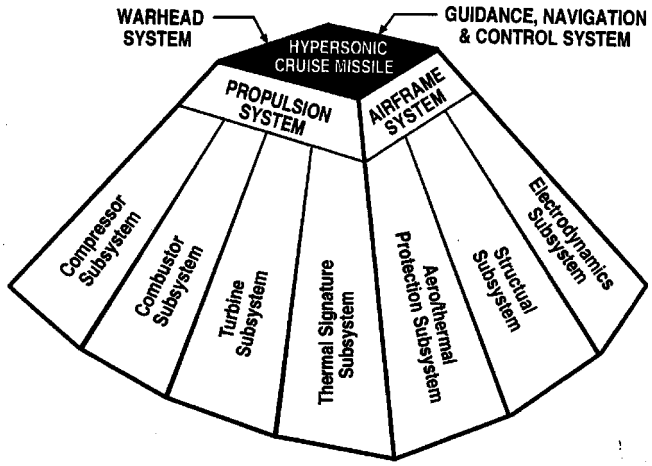


Fig. 11 Validation pyramid for a hypersonic cruise missile [11]

2.4.3 Characteristics of validation experiments

Many researchers, analysts, and managers ask: What is a validation experiment? or How is a validation experiment different from other experiments? These are appropriate questions. We suggest that traditional experiments could generally be grouped into three categories. The first category comprises experiments that are conducted primarily to improve the fundamental understanding of some physical process. Sometimes these are referred to as physical-discovery experiments. The second category of traditional experiments consists of those conducted primarily for constructing or improving mathematical models of fairly well-understood physical processes. The third category of traditional experiments includes those that determine or improve the reliability, performance, or safety of components, subsystems, or complete systems. These experiments are commonly called ''tests'' of engineered components or systems.

We argue that validation experiments constitute a new type of experiment. A validation experiment is conducted for the primary purpose of determining the validity, or predictive accuracy, of a computational modeling and simulation capability. In other words, a validation experiment is designed, executed, and analyzed for the purpose of quantitatively de-

termining the ability of a mathematical model and its embodiment in a computer code to simulate a well-characterized physical process. Thus, in a validation experiment ''the code is the customer'' or, if you like, ''the computational analyst is the customer.'' Only during the last 10 to 20 years has computational simulation matured to the point where it could even be considered as a customer. As modern technology increasingly moves toward engineering systems that are designed, and possibly even fielded, based on modeling and simulation, then modeling and simulation itself will increasingly become the customer of experiments.

In our view, there are three aspects that should be used to optimize the effectiveness and value of validation experiments. The first aspect is to define the expected results of the experimental validation activity using the code itself. The second aspect is to design specific validation experiments by using the code in a predictive sense. The third aspect is to develop a well-thought-out plan for analyzing the computational and experimental results. These aspects emphasize tight coupling of the subject code to the experimental activity and ensure we are in the best position to learn from the comparison of computational and experimental results, regardless of whether these comparisons are good or bad, as explained below.

Consider the second aspect above related to the design of a validation experiment. Suppose, through a series of exploratory calculations for a particular application of the code, that an unexpectedly high sensitivity to certain physical parameters appears. Specific PIRT elements for the application may have postulated little sensitivity to the parameters, or the PIRT may have stated that the sensitivity was unknown. If this unexpected sensitivity has an important impact on the application of interest, a validation experiment assessing the correctness of this sensitivity should be pursued. The code that is the object of the validation exercise should participate in defining the principles and goals of a planned validation experiment. An alternative phrase to keep in mind when the word ''define'' appears is ''define the purpose of the experiment.'' The role of the code in the definition of validation experiments may be purely exploratory until some progress has been made on validation activities.

One way of designing a validation experiment is to use the code calculations as specific guidance on where to locate the instrumentation and what kind of data to acquire for assessing the anticipated sensitivity. In the proposed validation experiments, ''design'' means specifying to the greatest degree possible the initial and boundary conditions, material properties, diagnostic locations and characteristics (strain gauge? temperature gauge?), and data fidelity. The probability that these conditions, ''defined'' through code calculations, will be met precisely in any complex validation experiment is probably zero, but the probability that they will be met in simpler experiments is relatively high. In most cases, the success of a validation experiment will often be determined by the degree to which the experiment matches these specifications. Deviations can be acceptable; but if the intent of the experiment was to measure x at location y for material z and geometry g and in all cases the experiment is significantly different from most or all of these factors, it is unlikely that the experiment would be a successful validation experiment.

Engaging a code in the definition and design of validation experiments can easily be distinguished from more common experimental practice. In the conventional case of nonvalidation experiments, the experimenters state the following to the computational analyst in the typical approach: ''We are performing the following experiment. Tell us what the expected response of gauge a will be, given this experimental design and location for the gauge.'' For a validation experiment, on the other hand, the computational analyst makes the following statement to the experimenter: ''Please perform the following experiment. We recommend locating gauge a at location x based on our computational predictions. It will be very useful to see whether our predictions agree with your experiment. If your experiment has deviated from our requested design, we still anticipate achieving useful validation consequences through further post-experimental analysis, as long as the design deviation is not great.''

A number of researchers, particularly experimentalists, have slowly been developing the concepts of a validation experiment. During the past several years, a group of researchers at Sandia National Laboratories has been developing philosophical guidelines and procedures for designing and conducting a validation experiment. Although the following six guidelines and procedures were developed in a joint computational and experimental program conducted in a wind tunnel, they apply over the entire range of computational physics [144–150]:

Guideline 1: A validation experiment should be jointly designed by experimentalists, model developers, code developers, and code users working closely together throughout the program, from inception to documentation, with complete candor about the strengths and weaknesses of each approach.

Guideline 2: A validation experiment should be designed to capture the essential physics of interest, including all relevant physical modeling data and initial and boundary conditions required by the code.

Guideline 3: A validation experiment should strive to emphasize the inherent synergism between computational and experimental approaches.

Guideline 4: Although the experimental design should be developed cooperatively, independence must be maintained in obtaining both the computational and experimental results.

Guideline 5: A hierarchy of experimental measurements of increasing computational difficulty and specificity should be made, for example, from globally integrated quantities to local measurements.

Guideline 6: The experimental design should be constructed to analyze and estimate the components of random (precision) and bias (systematic) experimental errors.

The purposeful design of validation experiments is enabled and accomplished through a diverse team of people who participate in the experimental validation activity. Most obviously, one or more experimentalists are participants in

this team. However, code users (analysts, designers) must also participate, given our proposed primary role of the code in the definition, design, and analysis of validation experiments. Finally, one or more code developers should be on the validation experiment team. Their presence provides valuable expert knowledge about the perceived *a priori* capability of the code in all three aspects of experimental validation: define, design, and analyze. In fact, code developers, including experts on the physical models in the code, are the most knowledgeable about the boundary of the acceptable application domain discussed in Fig. 9.

Validation experiments should not produce data that fundamentally depend on code calculations for critical data-reduction tasks. Such data do not correctly address our need for independent comparability of experimental data with calculations and violate our need for robustness of the experimental data. Experimental data that require code calculations for evaluation can never be a desirable outcome for a validation experiment, although exactly this problem may arise in other kinds of experiments. An example of what we mean is the problem of inference of material temperature from experimentally acquired shock hydrodynamic data (density, pressure, and velocity fields) using code calculations of shock-wave physics rather than some type of diagnostic to directly measure the temperature. The only possible validation data that will emerge from shock hydrodynamics experiments without temperature diagnostics are the shock hydrodynamic data. This problem is relevant, since it arises in investigations of temperature dependence in high-pressure shock-driven material response. Such experiments often need code calculations to estimate the associated thermal conditions under shock loading. For purposes of scientific discovery, this is permissible though speculative. Such experiments, however, cannot be claimed to provide validation data for the high-pressure thermomechanical response of the material because of the lack of independence of calculations and experiment.

Attempting to sharply define the boundary of applicability of the code for a given application domain through a deliberate choice of experiments close to the boundary, but on the invalid side, has greater leverage when code predictions are used to design such experiments. We stress once again that achieving a satisfactory level of performance for the code in comparison with validation experimental data in a case of true prediction has far greater power to raise our level of confidence in the application of the code. We may not be able to quantitatively measure this fundamental observation at this time, but it is obviously true. Predictive accuracy is gold, while post-test consistency is merely brass at best and possibly fool's gold at worst.

2.4.4 Uncertainty quantification in computations

As mentioned in Section 2.4.1, it is common when simulating validation experiments that one encounters physical parameters, eg, in the PDEs or in the initial or boundary conditions, that are not precisely known or measured for an experiment. This situation is much more common at the system and subsystem tiers than at the benchmark and unit-problem tiers. Examples of such parameters are thermochemical transport properties, flow rates in complex systems, and inflow nonuniformities. Another common situation occurs during a series of experiments, eg, weather-driven experiments and flight tests, when there are certain parameters that are poorly controlled or not controllable at all. For any parameter, we make the assumption that a value of that parameter is required to perform the computational simulation. The situations just mentioned are all characterized by stochastic parametric uncertainty. In the following discussion, we simply refer to ''uncertainty,'' but we will always be referring to parametric uncertainty. This type of uncertainty requires nondeterministic simulations in validation of computational physics.

One standard approach is to estimate, by one means or another, a single value of such a parameter and compute a solution with that selected value. This might be a fully adequate way of dealing with this uncertainty, especially if experience suggests that the range of potential parameter values is very small and that the calculation is known not to be extremely sensitive to the parameter in question. The resulting calculation intrinsically is interpreted as ''representative,'' or ''best estimate'' or ''conditional'' for that parameter.

The shortcomings with the standard approach referred to above begin to be noticeable when the range of variation of the parameter is large or when the calculation is known to be sensitive to the parameter values. If multiple required parameters in a computation of a validation experiment are uncertain, it is entirely possible that the interaction of these parameters in the calculation may magnify the influence of their uncertainty on the final results of the calculation. In our opinion, this statement is especially important when performing calculations that are intended for direct quantitative comparison with validation experiments. We believe that the uncertainty of the parameters should be incorporated directly into the computational analysis.

The simplest strategy for incorporating uncertainty of this kind directly into the computation is performed in three steps. The first step, called *characterizing the source* of uncertainty, is based on the assumption that the uncertainty in the parameters of interest is characterized by probability distributions. Sometimes such distributions can be directly estimated if a large quantity of experimental data is available for the parameters. Sometimes such distributions simply must be assumed. At any rate, the first step involves specifying the probability distributions and understanding the credibility of these assumptions.

In the second step, *ensemble computing*, values from the input probability distributions specified in the previous step are selected using statistical sampling procedures, such as Monte Carlo or Latin Hypercube sampling methods (see, for example, [41,151]). These sampled values are then used in a set of computations. Because this latter statement is so important, we will restate it. The assumed prior probability distributions for the uncertain parameters are used to generate a set of calculations. This set is sometimes called an ensemble of calculations.

The key issue is that a single calculation is no longer

sufficient; a set of calculations must be performed. Obviously, this need is disturbing; where once one might have performed a single calculation, now one must perform a potentially large number of calculations. We have not raised nor answered the question of whether sufficient computational resources are available to execute more than the one calculation. However, the constraints enforced by the availability of computing may be formidable.

After the set of calculations has been generated, the third step, *uncertainty quantification of the output*, is performed. This step involves analysis of the set of calculations, typically using statistical inference, to estimate a probability distribution for the output variables of interest that results from the given input parameter distributions. In general, we cannot deduce the exact output distribution that results from the assumed form of the parameter distributions used to generate the computational input associated with those parameters. Instead, the common practice is to use statistical procedures to determine estimates of important parameters associated with that output distribution.

Such statistically determined estimates are useful for comparing computational results with experimental measurements. For example, the mean of the output calculations provides us with an estimate of the expected value of the output quantity of interest, given the uncertainty structure specified by the input distributions. This mean-value estimate is of primary importance when comparing computational results with the mean value of multiple experimental realizations. Another statistic of interest is the estimated variance of the output distribution, which can be interpreted as a measure of computational output uncertainty, or scatter, given the input uncertainty.

For readers unfamiliar with this methodology, we stress that it is rarely true that the mean of the output, given the input uncertainty, can be determined by performing a calculation for a single set of inputs that is chosen to be the mean of each of the input distributions. Stated in another way, the mean value of the output cannot be computed by performing a simulation using the mean value of all input parameters, except when the mapping of inputs to outputs is linear in the parameters. Linearity in the parameters essentially never occurs when the mapping of inputs to outputs is given by a differential equation, *even a linear* differential equation. Instead, we must perform the ensemble of calculations to develop a statistically rational estimator for the mean. The previous statement is also true for estimating other output statistics. Kleijnen [152] provides a broad summary of methodologies that go beyond Monte Carlo for assessing output distributions statistically.

In summary, the general methodology we are thus advocating for incorporating parameter uncertainty into computational physics computations is to execute all three steps in the manner suggested above. Performing the three-step strategy will clearly be nontrivial for hard computational problems simply because of the computational burden imposed. There are also certain subtleties that we have not mentioned, such as whether complex structure in the resulting output probability distribution can actually be discovered using such a crude approach. Extracting an intricate output-distribution structure will require either a large number of sample input values or considerably more sophisticated approaches for performing the methodology. The only modeling and simulation applications we are aware of within computational physics that pursue this methodology are the fields of nuclear reactor safety [49–52], underground transport of toxic waste materials, or pollutants [153–156], probabilistic structural dynamics [157–159], and climatology [160–165].

We now return to our mention of Bayesian inference. In step two of the three-step strategy, the problem of propagating input uncertainty through a computational model to understand the resultant output as described above is sometimes referred to as the *forward uncertainty problem*. There is an associated *inverse uncertainty problem*, or *backward problem*, which is conceptually and mathematically much more difficult [166]. The backward problem asks whether we can reduce the output uncertainty by updating the statistical model using comparisons between computational and experimental results. The inverse uncertainty problem is a calibration problem, but in the inverse uncertainty problem the parameters being calibrated are given by probability distributions. For example, might we be able to improve our original prior distributions that characterize the parameter uncertainty? This problem can be cast as a problem in Bayesian statistical inference. (See [167–169] for an introduction to Bayesian inference.)

Part of the difficulty alluded to above is related to providing a better understanding of computational accuracy when it is known that we are in an underresolved grid or time-step situation. This is an important research topic because of its practical consequences. Recent work attacks this problem and illustrates the formidable difficulties in two distinctly different areas: fluid dynamics [134,170] and dynamical systems [131–133]. While the applications and specific technical attacks of researchers in these two areas are distinctly different, we find it fascinating that a common deep thread in their work is the treatment of insufficient information using statistical methods. We should stress, however, that these authors do not discuss one particular problem of interest to us—that of validation of underresolved computational models. If reliable estimates of error due to underresolved grids can be made, then we believe model validation can be conducted. The disadvantage, however, is that if large error bounds are estimated, there can be a great deal of room for the computational results to agree or disagree with the experimental data. Stated differently, only *weak* validation conclusions can be drawn if either the computational or experimental error estimates are large. If *no* estimate of grid convergence error can be made for the computational solution, we strongly believe *no* validation conclusion can be made. The same argument can be made if only one experimental measurement is made. However, it is traditional in engineering and science that relying on only one experimental measurement is more acceptable than not estimating grid convergence error, regardless of whether this position is actually defensible. The issue of how experimental uncertainty should be treated in validation is addressed in Section 2.4.5.

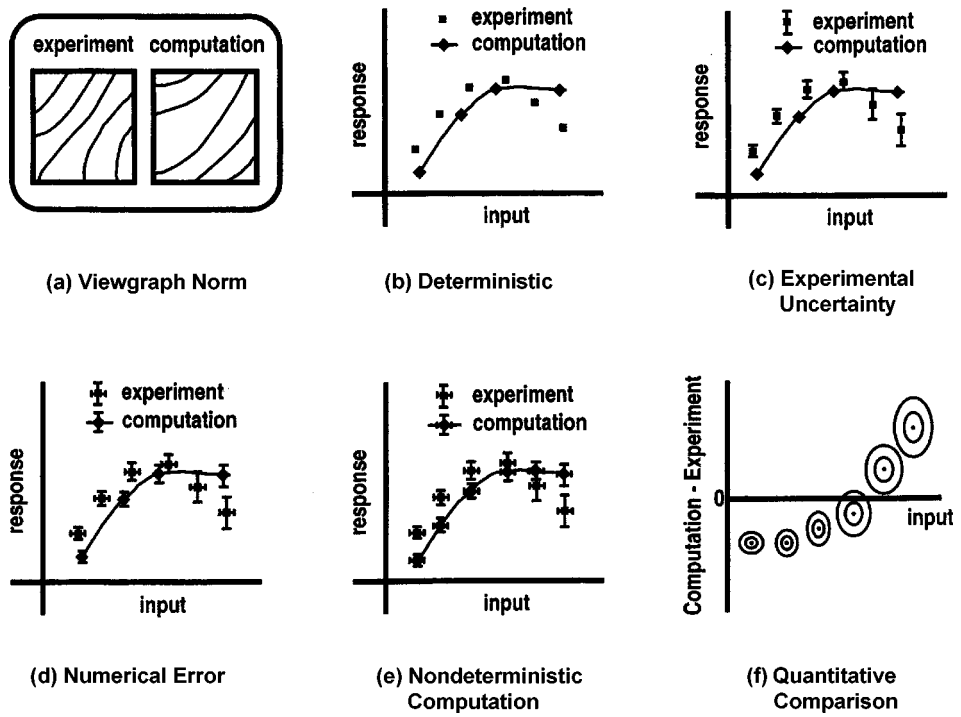


Fig. 12 Increasing quality of validation metrics (adapted from [30])

2.4.5 Validation metrics

The specification and use of validation metrics comprises the *most important* practice in validation activities. Validation metrics are used to quantitatively compare the results of code calculations with the results of validation experiments. The straightforward interpretation of this word is that a ''metric'' is simply a ''measure.'' Thus, the choice of one or more metrics defines the means used to measure the differences between computational results and experimental data. Because we emphasize that the overarching goal of validation experiments is to develop sufficient confidence so that the code can be used for its intended application, we do not view the use of metrics as simply a passive means of measuring differences between computational results and experimental data. Metrics must be devised that actively resolve assessment of confidence for relevant system response measures for the intended application of the code. Referring back to Fig. 5, it is clear that validation metrics are important for situations portrayed in Figs. 5*a* and 5*b*, but they are *crucial* for quantitatively estimating confidence in modeling and simulation capability for Fig. 5*c*. The definition, design, and application of these metrics should directly flow from the PIRT activities described in Section 2.2.

A metric should quantify both errors and uncertainties in the comparison of computational results and experimental data. Specific examples of using metrics can be found in a number of references [11,31,171–189]. Although there is a large amount of recent work in this topic, it is our view that this area of validation quantification techniques is in its infancy, despite its great importance.

Figure 12 helps to clarify where we believe the present state of the art is in computational-experimental comparisons and to identify the direction in which we believe the topic needs to progress. This figure illustrates the conceptual increase of quantification in performing validation comparisons as increased attention is paid to both experimental uncertainty and computational error. The figure depicts a desirable progression from *a* qualitative comparisons in which experimental data and calculations are presented side by side on a viewgraph without any information about uncertainties in either one, to *f* analysis of quantitative differences between experiments and calculations accounting for uncertainty and error. Let us consider this figure in some detail.

Figure 12*a* depicts a qualitative comparison we refer to as a ''viewgraph norm'' comparison of experimental and computational data that is often seen in practice. This comparison is marked by the qualitative nature of the chosen metric, typically comparison of color contours over a domain. It is also distinguished by no information on experimental and computational uncertainty. No quantitative statement about validation can seriously be made based on such a comparison, although the statement may provide some feel of confidence in the application of the code at an intuitive level. Intuition and feel of agreement between computational results and experimental data, of course, are in the eyes of the beholder, and not necessarily either traceable or reproducible.

The plot in Fig. 12*b* portrays the most common type of comparison between computational results and experimental data. Generic axis labels of system ''input'' and ''response'' are shown, although both axes can also be system response quantities over a range of an input quantity. While discrete experimental and computational points are shown in this

plot, the concept also encompasses curve overlays without any points shown. The key problem with metrics implemented at the level of Fig. 12*b* is that there is no recognition of uncertainty in the comparison. Conclusions drawn from this type of comparison are really only qualitative, such as ''fair agreement'' or ''generally good agreement.''

Figure 12*c* suggests that the next step for improving the method of comparison is to place qualitatively estimated error bars around the experimental data. By ''qualitatively estimated error bars'' we mean the common practice of the experimentalist either a) quoting the manufacturer's stated accuracy of the diagnostic sensor, b) making a statement such as ''Essentially all of the experimental data fell within the error bars shown,'' or c) ignoring experimental uncertainty in the input, or control, quantity. At this stage of comparison a validation metric could be constructed that accounts for two characteristics: the difference between the computation and experiment integrated over the range of the input parameter, and some estimate of the experimental uncertainty.

Figure 12*d* represents the case where there is a more quantitative estimate of experimental uncertainty and there is an estimate of computational error. The information concerning experimental uncertainty could be improved, for example, in two ways. First, multiple experimental realizations could have been obtained so that the experimental data point shown would represent the mean of all of the samples, and the error bar would represent two standard deviations of an assumed Gaussian probability distribution. Second, an estimate of the experimental uncertainty in the measured input quantity is also obtained and is shown as an error bar. Concerning the computational error, an *a posteriori* numerical error estimate from the computation would be given for the specific system response quantity that was measured. The computational error estimate may have come from, for example, a single-grid solution or an estimate based on solutions from multiple grids.

Figure 12*e* suggests a further improvement in estimation of experimental uncertainty and also computation of nondeterministic simulations. For example, the experimental uncertainty could include an estimate of correlated bias errors using the procedure discussed earlier, or an estimate of uncertainty based on the same experiment conducted at separate facilities. Concerning the nondeterministic computational simulations, we are referring to an ensemble of computations at each of the experimental conditions, as discussed earlier. For example, multiple computations would be made using the experimentally estimated probability distribution for the input quantity. As a result, the computational data point would be the mean of the nondeterministic computations for both the input and the system response. The ''error bars'' for the input quantity and the system response would represent two standard deviations for each of the computational probability distributions. Note that to use the experimental uncertainty distributions for the input quantity, computations would need to be made at the measured input conditions.

Figure 12 *f* shows a true quantitative measure of the comparison between the computations and the experimental measurements over the range of the input quantity. In terms of information content, one could have the same data as contained in Fig. 12*e*. However, we believe a validation metric should be based on the difference between computations and experiments over the range of measurements of the input parameter. Assuming that probability distributions for both computational and experimental data are known, as discussed in Fig. 12*e*, comparing computations and experiments will require a difference, or more properly, a convolution of pairs of probability distributions. (See, for example, [190,191] concerning convolutions of distributions.) The elliptical symbols in Fig. 12 *f* are meant to signify one and two standard deviation contours of the convolutions of the computational and experimental probability distributions. The ''dot'' in the center of each contour is the difference in the mean, or expected value, of the computational and experimental distributions.

As proposed in [11,12], we believe that all reasonable validation metrics should include a number of useful properties, although validation metrics need not be restricted to satisfying only these properties. The following is a list of properties that we believe a validation metric should have:

1) A metric should incorporate an estimate of the numerical error in the computational simulation. This estimate may only be an estimate from solutions on underresolved grids. However, we believe that representing the uncertain estimate of error as a probability, eg, 1 out of 100 chance, is not defensible, since convergence of numerical error is more closely analogous to bias errors in experimental measurements.
2) A metric should not exclude any modeling assumptions or approximations used in the computation of the simulation result. A metric should reflect all uncertainties and errors incurred in the modeling process.
3) A metric should incorporate estimates of the random errors in the experimental data that are the basis of comparison. In addition, we believe a metric should also include an estimate of the correlated bias errors in the experimental data, if possible.
4) A metric should depend on the number of experimental replications of a given measurement quantity. That is, a metric should reflect the level of confidence in the experimental mean that has been estimated, not just the variance or scatter in the data.
5) A metric should be able to incorporate computational uncertainty that is due to both random uncertainty in experimental parameters required for defining the calculations and any uncertainty that is due to lack of experimental measurement of needed computational quantities. Thus, a metric should use nondeterministic methods to propagate uncertainty through the subject computational model.

The importance of quantifying the experimental uncertainty has been emphasized in the discussion of experimental design above. Quantifying experimental uncertainty is crucial for validation metrics. The inference from validation experiments to the application domain as discussed in Section 2.1 is greatly weakened in the absence of quantifying the

experimental uncertainty. Similarly, it is critically important to estimate numerical errors in the computational simulation that are compared with the experimental results. Otherwise, in computational-experimental comparisons, the numerical errors can be indistinguishable from the modeling uncertainties.

It is also important to properly represent the uncertainty observed in the experiment. For example, observed experimental variations can be propagated through ensembles of calculations using techniques for propagating uncertainty, and the computational results can then be analyzed to provide statistical data about the resulting computational uncertainty. In practice, however, difficulties arise for several reasons. First, the raw effort involved in obtaining ensembles of calculations to propagate uncertainty becomes highly taxing for complex computations. Second, it is very difficult to accurately assemble the experimental uncertainty data into precise statistical information for the routine propagation of computational uncertainty. The inability to accurately characterize experimental uncertainty as probability distributions may be due to, for example, inadequate diagnostic information on physical parameters or to limited time and financial resources. When information is lacking, referred to previously as epistemic uncertainty, some of the newer methods in information theory could be used. However, these methods are not well developed for practical engineering problems. Even with these difficulties, we believe the definition and use of validation metrics can still proceed even if information about computational uncertainty is lacking or poor. In effect, when a poor job is done in quantifying the computational uncertainty, the quality of the conclusions that can be drawn from the validation metrics is adversely affected. However, when a poor job is done in quantifying the experimental uncertainty, the goal of the validation experiment can be defeated.

Comparing an ensemble of calculations with an ensemble of experimental data is a statistical problem of quantifying uncertainty. Statistical analysis of the differences between computational and experimental results must play a key role in the analysis and presentation of metric data. A fundamental goal of a validation metric is, then, to present at least the impact of the uncertainty of the experimental data upon the inferences that are drawn from the validation exercise. It is important to recognize that uncertainty in the experimental data (and of the computations) affects the credibility of the results of a validation metric. This fact, in turn, will impact the confidence in the proposed application domain of the code.

3 MAJOR RESEARCH ISSUES

3.1 Prioritization of assessment activities

In this section we advocate widespread use and further research into the use of the PIRT for prioritizing V&V activities and explain our reasons for doing so. The use of the PIRT for this purpose is fairly well developed in the field of reactor-safety risk assessment; however, its use in other fields has been very limited. We believe the PIRT is the only method presently available to account for the needs of the intended application, the believed present capability of a code, and areas where the code is questionable. When these factors are combined, with prioritization of multiple intended applications, one is able to identify and prioritize the necessary V&V activities. In our use of the PIRT in the ASCI program at Sandia National Laboratories we have learned some important lessons that can inform future practice in V&V. The first lesson learned is that when the intended applications of a code are being identified and prioritized, the process requires close interaction between the computational analysts and the ''customers'' of the computational analyses. The customers may be design engineers, safety engineers, reliability engineers, and project engineers who would all typically serve as decision makers about engineering-related issues. Without such interaction, computational analysts can lose touch with the perspectives and priorities of their analysis customers. The second lesson learned is that the PIRT can uncover requirements for computational simulations that had not been recognized as important or that had not been recognized at all. For example, the PIRT can be particularly helpful in identifying important unrecognized areas of coupled physics, either on one face of the validation pyramid or coupling between different faces of the pyramid (see Fig. 11).

We also learned that the PIRT can be helpful in identifying a spectrum of different types of experiments and tests that are likely to be needed in the future. The following types of experiments and tests have been identified [30]:

- Phenomena-exploration experiments
- Mathematical-model-development experiments
- Calibration (model-updating) experiments
- Validation experiments
- System and certification tests

The key difference between this spectrum and the hierarchical validation experiments discussed earlier is the goal of each: validation experiments focus on the *assessment* of modeling capability, regardless of the level of physical complexity.

This spectrum of experiments influences the development and growth of confidence in computational models and their applications in an evolutionary sense, as depicted in Fig. 13. In this figure a serial view of the role of the above experiments is emphasized, as well as the interaction of experiments and the code. In the left column we cast our view of experimental activities in an ordered sequence consisting of the five types of experiments: phenomena exploration, mathematical model development, calibration, validation, and system and certification tests. In reality, of course, there are possibly local loops that have not been emphasized in Fig. 13 that further connect each of these experimental elements. For example, it is apparent that the conduct of experiments that are designed to be validation experiments could reveal the need to perform experiments from any other element in the column.

Each element in the experimental domain directly interacts with an element in the computational domain. The motivating concepts generating these computational elements

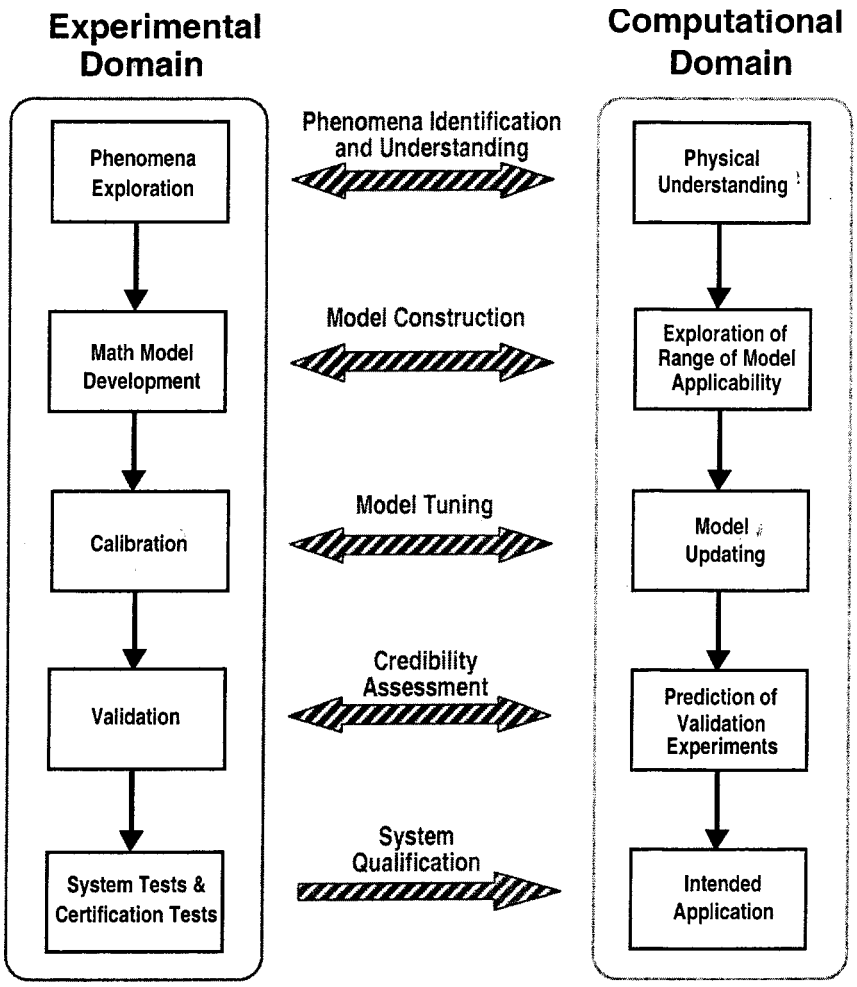


Fig. 13 Interaction of various experimental and computational activities (adapted from [30])

are suggested by the arrows in the middle of the figure that depict the flow of experimental information directly to the computational domain and vice versa. We have simplistically characterized the computational domain through the types of activities expected to be most associated with the directly correlated experimental element. We believe that our depiction in Fig. 13 highlights the right connections between the experimental and computational domains if the experimental domain is not restricted only to validation experiments as they are defined in this report. In particular, the ultimate goal of both experimental and computational activities is to provide a basis of confidence and decision making for the intended application of the code. We have highlighted the great weight that system tests and certification tests evoke by emphasizing a more unidirectional flow of information between the experimental and computational domains at this level. The technical weakness to their genuine physical reality, if the system can be tested, is that the test is only one physical realization of a complex, nondeterministic system.

3.2 Verification activities

This section identifies areas of improvement that can help advance the effectiveness of the verification process in computational physics. We recommend the development of standardized test problems, the further development of MMS, and the use of statistical methods for the quantification of SQA. The issue of verification in chaotic systems is also briefly addressed.

Previously we have argued for the special importance that highly accurate solutions play in numerical algorithm verification. For brevity, we will simply refer to highly accurate solutions as ''benchmarks.'' These benchmarks can be analytical or numerical solutions. There are two senses in which we mean benchmarks for use in algorithm testing. Weak-sense benchmarks are test problems that are in common *ad hoc* use by various algorithm developers or code projects for the purpose of assessing numerical accuracy. We define strong-sense benchmarks fundamentally to be *engineering standards*. We will discuss strong-sense benchmarks in Section 4.4.

We believe using benchmarks constructed by MMS are the most effective technique for numerical algorithm verification. Although many code developers question the effectiveness of MMS, we have found that everyone who has used the technique is now a strong believer in its power. Users of the method have demonstrated that it is extraordinarily sensitive to coding errors of the numerical algorithms: as Roache puts it ''annoyingly so.'' In fluid dynamics, MMS

benchmarks should be developed, for example, in supersonic flows with shock waves, turbulent flows, reacting flows, multiphase flows, non-Newtonian flows, free-surface flows, and large-eddy-simulation models. We recommend that MMS benchmarks be developed and applied to other areas of physics, for example, radiation transport, large plastic deformation, fracture dynamics, electromagnetics, and quantum mechanics. Two additional topics of MMS that require further research, for all areas of physics, are 1) correct imposition of boundary conditions for a mixture of elliptic, parabolic, and hyperbolic systems; and 2) determination of types of code errors that are *not* detectable by individual applications of the MMS. Stated differently, the limits of the MMS should be investigated more thoroughly so that any redundancy with traditional analytical-solution verification testing could be better understood.

Statistical testing methods for the quantification of SQA are becoming more widely used. We recommend that these methods should be developed more thoroughly, primarily by research teams composed of computer scientists and statisticians. We believe that statistical testing methods can be effectively used in the computational engineering and physics community; however, we are not aware of any studies to date. The recent text by Singpurwalla and Wilson [192] is a good introduction to the topic of software reliability and detailed methods for statistical quantification. To contrast the view of statistical testing of software with the view expressed in the present paper, we note that the book by Singpurwalla and Wilson does not use the term ''verification'' anywhere in the Table of Contents or in the Index.

The final research topic in verification activities that we discuss in this paper goes beyond the topic of error estimation. We can ask fundamental questions of how well the computational world matches the ''true'' dynamics of the underlying equations. For example, how well are the dynamical system features of the underlying differential equations matched by the computational dynamics? In the case of steady flows, this question is concerned with the stability and bifurcation behavior of the computed solutions. Attempting to measure the computational ''error'' in this case is very difficult. For example, it is only in relatively simple cases that the ''true'' stability and bifurcation behavior may be well understood for the conceptual model. An entire taxonomy of potential threats to computational accuracy arises in this point of view. Potential dangers are summarized in [193] and the work cited there.

3.3 Validation and predictive activities

In this section we identify areas of improvement that can help advance the effectiveness of the validation process in computational physics. We recommend the construction and use of a validation hierarchy and the specification and use of quantitative assessment criteria for validation metrics at the different tiers of the hierarchy. Several suggestions are also made regarding the need for developing additional validation metrics based on statistical measures, as well as those that are applicable to a wider range of physical processes.

The construction and use of a validation hierarchy for complex engineering systems is relatively new, but we believe this approach will be fundamentally useful in the future. In fact, we believe it is the *only* constructive approach for decomposing complex systems that can build demonstrable confidence in individual computational models. Consequently, we recommend that organizations that use computational modeling in the design, certification, or production of engineered systems should begin constructing and using the validation hierarchical structure. Admittedly, effective validation hierarchies—those that emphasize single-physics phenomena at the lower levels and engineered system-performance priorities at the higher levels—are difficult to construct. These hierarchies can only be constructed in an iterative fashion with input from a wide range of professionals, from engineering analysts to design engineers to project-level decision makers. In our limited use of the approach, we have found that a validation hierarchy quickly points out areas where little or no validation evidence exists. The ability of this approach to elucidate insufficiency or weakness in the validation evidence may not be appealing to some advocates of computational simulation, but knowledge of such deficiencies is essential from an engineering perspective.

Validation experiments conducted at the higher tiers of the hierarchy (subsystem and system) will invariably have poorly characterized experimental data for input to the computational simulation. As a result, validation will necessitate probabilistic treatment of uncertain parameters in the computational physics submodels or in the initial conditions or boundary conditions for the PDEs. Propagation of these parameters or conditions through the computational physics model will likely rely on probabilistic sampling methods like Monte Carlo or Latin Hypercube sampling. Importantly, major increases in computational resources will be required for the tens or hundreds of solutions needed for the sampling techniques. Additionally, improved training and understanding for probabilistic sampling techniques will be required for the computational physics analysts involved in statistical validation activities. On this topic, we believe that the computational physics community can learn a great deal from the probabilistic structural dynamics community and the probabilistic risk assessment community and can use computational tools developed by these communities.

We believe the validation team must be able to assess the results of the validation metrics. This activity is referred to as ''Metrics Assessment'' (block 6) in Fig. 6, presented previously. The primary goal of such an assessment is to credibly establish whether the agreement of calculations with experimental data satisfies the application requirements. This is the key purpose of dedicated validation experiments, and it is what distinguishes validation from a straightforward computational simulation of an experiment. Failure to assess the validation metrics with respect to the intended application makes it virtually impossible to understand what information about model confidence has been generated as a result of the validation activity. In conventional practice, assessment criteria for validation metrics are typically formulated, if at all, after the experimental data have been collected. We believe it is very useful to define some of these assessment criteria,

assuming this can be done, during the ''Intended Application'' activity and the ''Planning'' activity (see Fig. 6). More detailed specification of the assessment criteria will be possible after the validation experiments are complete, but it will still be difficult.

Specifying the assessment criteria for validation metrics will be difficult for two reasons. First, application requirements for the accuracy of computational analyses are commonly very vague. When application requirements are given, they are typically given for relatively high-level measures of system response. For example, a system-level requirement for debris containment during the catastrophic failure of a gas turbine engine may simply be that all engine-part debris must be contained within the engine shroud. Second, if the application requirement gives little or no guidance for detailed system-response measures, how should these assessment criteria be determined? For example, in the case of debris containment described above, how should the application requirement be consequently determined for models for metal tearing or large plastic-deformation metal folding?

Two approaches are recommended for guidance in determining the assessment criteria at the different tiers that are shown in Fig. 10, presented previously. First, the metrics assessment activity (identified in Fig. 6) is not intended to completely answer the question of whether the code is suitable for the intended application. Clearly, the intended application should dominate the criteria for assessing the results of validation metrics. The fact remains that the quality of comparisons of computational and experimental results can also be judged somewhat independently of their ultimate application, purely as a problem in a scientific area. People who historically engage in computational modeling of experiments make these judgments as a matter of course, but rarely are these judgments quantified or formalized to the degree being advocated for validation purposes in this paper. Second, guidance for assessment criteria for validation experiments done at the unit-problem tier and the benchmark tier should also be derived from sensitivity and uncertainty analyses done at both the subsystem tier and the system tier. Stated differently, the impact on the system tier of given levels of accuracy at the lower tiers should be determined with sensitivity and uncertainty analyses. Although these analyses are hard to accomplish early in the development of a code, they will need to be done during the intended application activity and the planning activity.

While the questions and concerns listed above for specifying the assessment criteria are significant, it is our view that the quantification of requirements at multiple tiers is the correct approach. When this approach is compared to the traditional test-based engineering approach, it is seen that the approaches are actually analogous. The test-based engineering approach, however, is more *ad hoc* at the lower tiers, and there is much more emphasis on testing of the full-scale system.

Making validation-metric success criteria as quantitative as possible is important. It is possible that success criteria may be qualitatively or indirectly defined, perhaps through a process of assimilated experience during the course of the validation activity. In our opinion, however, qualitative criteria, or criteria that are developed after the activity of comparing computational and experimental results has begun, make the validation job more difficult and undermine the credibility of the assessment. In addition, qualitative criteria that are developed after the execution of experimental validation activities lead to the danger of focusing the definition of success on discovering any feature in the comparison that is ''good'' at the expense of answering the harder question: Is there an important reason that this feature in the computational-experimental comparison should be good?

Additional research is needed to define validation metrics for a wide range of system-response measures. For example, in fluid and solid dynamics we note that validation metrics are needed that weight important regions of the domain more heavily than less important regions. Metrics are also needed for unsteady continuum mechanics. If the computational solutions are periodic or quasi-periodic, we believe that validation metrics should be formulated in the frequency domain, not in the time domain. As is well known from structural dynamics, the frequency domain eliminates the unimportant feature of phase shifts in the periodic response and concentrates on natural modes in the system. For example, Urbina and Paez [186] have focused development of validation metrics for structural dynamics models using frequency domain metrics, in particular a windowed spectral response function metric. Apparently random, possibly chaotic, temporal acceleration histories at various locations in complex structural systems result from close, but not solid contact, between structural elements. We observe that the application of frequency-domain validation metrics in complex fluid flows has some relationship to certain techniques that have been developed for assessing prediction uncertainty in climate modeling using singular-value decompositions [164]. For other kinds of flows, of course, elaboration of complicated time-domain metrics may still be important. For example, in an oil reservoir flow problem Glimm *et al* [194] devise a complex time-domain metric that is used to predict confidence intervals for simulation predictions of reservoir production given a stochastic geology.

Another research problem that needs vigorous attention is to formalize the propagation of validation metric information from the unit problem/benchmark tiers in the validation hierarchy to the subsystem/system tiers. We mentioned above that this problem is related to sensitivity analysis and propagation of uncertainty in the validation hierarchy. What makes the problem even harder is that second order uncertainty is present in the validation hierarchy. That is, quantitative specification of information-flow within the hierarchy is itself highly uncertain, at least for strongly-coupled-physics problems. While it may be the case that nontraditional uncertainty quantification methods are required to address this problem, researching and applying probabilistic methodologies are certainly important to begin with. Recently, Hills *et al* [195] have begun a study of the propagation of validation metrics and their associated uncertainty within a simple validation hierarchy using first-order-sensitivity methods and a maximum-likelihood-based validation metric. The problem

of propagation of validation metric information in a validation hierarchy also has substantial formal similarity to the problem of system reliability methods. Recent work applying Bayes net methods originally developed for system reliability applications to understanding system validation metrics has just begun [196].

Finally, we believe that improvements in validation metrics can be gained by adoption of some of the methods used in model calibration. Validation puts a premium on obtaining a measure to compare computational and experimental results and relating the magnitude of this measure to the intended application. Calibration emphasizes methods to improve the agreement between the model and the experiment by optimization of model parameters. It is clear that metrics used in the first step of calibration, ie, before optimization is introduced, are directly pertinent to validation metrics discussed here. After a model is calibrated or updated, the same question remains: ''How do we quantify the confidence in our calibrated model?'' A recent review of some of these issues with a focus on calibration and Bayesian updating is found in [197]. Bayesian model updating requires a probabilistic validation metric for improvement of probabilistic parameters in the model. Hasselman *et al* [198,199] has developed a different approach to compute probabilistic validation metrics for structural dynamics analyses. He uses principal components analysis and singular-value decomposition in the formation of the metric.

4 MAJOR IMPLEMENTATION ISSUES

Many of the research issues just discussed will require mathematical methods development and careful thought and debate over a number of years to resolve. We believe there are also a number of other issues that are just as important for progress of V&V, but they are issues related to management and deployment. In this section we discuss: requirements for improved understanding of differing perspectives, new management initiatives, new ''customer'' initiatives for using commercial software, and the development of engineering standards for code verification. Without attention, several of these implementation issues may actually be more detrimental to the maturation of trust and confidence in computational simulation than the lack of research progress.

4.1 Management issues

In this section we address implementation issues related to the need for significantly improved code verification, the need for improved cooperation between computational analysts and systems engineers, the need for synergism between analysts and experimentalists, and the need for more quantitative V&V indicators. All of these issues present difficult management challenges to transform the status quo. We note that the issues discussed in this section predominantly relate to organizations that develop their own computational physics software or organizations that use commercial software in their activities.

The growing impact associated with computational physics software failure, the increased size and cost of code projects, and the need for interaction of large numbers of software developers greatly leverage the involvement of, and reliance upon, formal SQA practices in the development of computational physics software. Resistance to the application of formal SQA practices in computational physics is created by the degree of formality, constraints, and seeming costs of SQA, as well as by psychological reasons (science versus product development, for example). As mentioned by Roache [14], SQA issues are downright unattractive to small computational physics projects whose primary purpose is to develop software for exploration and scientific insight, rather than to minimize the risk of software failure at seemingly great cost.

Significantly, as the goals and expectations of computational physics evolve to have impact that goes far beyond scientific exploration and insight, the consequence of software failure greatly magnifies. In particular, inaccurate answers rather than obvious code failures are especially dreaded in high-consequence computing because it may be very difficult to determine that a calculation is inaccurate or misleading. SQA methodologies provide some additional means of addressing such a problem, but does the cost justify intensive application of such methodologies for a given computational-physics-code project? Part of the confidence-optimization problem for verification is the difficult question of how to measure the consequence of failure in a computational physics code. Consequence measures could involve, for example, potential loss of future business, possible liability costs (both human costs and property costs), magnitude of environmental damage, and national security impact. Such a measure is important in clearly understanding when the application of formal SQA methodologies provides unquestioned value in code verification activities.

Two widespread difficulties are encountered when applying computational physics predictions to real systems-engineering decision making. The first difficulty concerns the radically different time scales for making engineering decisions and producing results from computational simulations; that is, engineering decisions need to be made rather quickly on design issues and there is not much time to run many computational simulations. For example, it is not uncommon for the design engineer to tell the analyst ''If you get the analysis to me by next week, I will use the results to make a design decision. If you don't, I will make the decision without your help.'' The second difficulty is related to the radically different value systems that are in play between analysts and systems engineers. In our opinion, computational analysts typically have an infatuation with physical detail, while systems engineers are concerned with high-level issues such as design optimization, manufacturing producibility, system robustness, and assessment of system safety and reliability. We believe that systems engineers are the ultimate customers of the code and should be the dominant players in their interaction with computational analysts. Systems engineers are responsible for designing and producing products, and the primary role of the codes is to help them make better decisions.

As we have discussed previously, a competitive or adversarial relationship between computationalists and experimen-

talists exists in many organizations. This type of relationship can be found at the individual level or between computational and experimental groups. It could be due to competition over organizational resources or could be the result of recognition of one group over the other. Management often does not recognize the problem, or if they do, tend to subconsciously ignore it. And even if there are not competitive or adversarial pressures, there is commonly a strong difference in cultures between computationalists and experimentalists. For V&V to be successful, it is imperative that management assess the state of the relationship between computationalists and experimentalists in their organizations and create opportunities for bringing these different individuals and groups together in cooperative and mutually beneficial efforts. Management must make it clear that the success of the validation team effort will benefit both groups equally, and that failure will be the responsibility of both groups. By ''success'' we mean the accuracy, quality, and timeliness of the computational and experimental efforts: *not* whether the agreement between computation and experiment was good.

The issue of how to succinctly communicate quantitative information concerning the status of V&V activities on a code or computational result is an important issue. There is a practical need for quickly transmitting this information, for example, to potential customers for commercial software, customers of computational analyses, decision makers, and policy makers. This is particularly important when this information must be transmitted at a meeting or seminar-type presentation. Logan and Nitta [200] have suggested an intuitive and effective method for transmitting this information using graphical symbols of a dial-face meter for code verification, model validation, and even numerical error estimation. They have suggested using a zero to ten scale on each V&V meter to quantitatively indicate the level of completion, maturity, or accuracy of the activity. The mapping of the numerical scale to various levels of completion, by necessity, must be qualitative. However, the same qualitative mapping would be used for each computational result in an ensemble of results, or computational capability exercised in the code. In that way the observer/decision maker could consistently see the relative maturity of different computations or capabilities. *Achieving consistency*, even on a relative and qualitative scale, is a major implementation goal for V&V. It would be expected that there could be significant differences of opinion on the mapping between numerical and qualitative descriptors between, for example, commercial software competitors. However, we believe Logan and Nitta's VER- and VAL-meters are an effective way to move V&V information toward a more quantitative basis and conspicuous stance when brevity is of the essence.

4.2 Practical applications in industrial settings

A critical issue in improving the implementation of V&V activities, especially validation, is to understand the particular needs of the customers of the computational models. From the perspective of a code developer, these customers may be analysts who use the models to produce results or they may be system engineers who use the results for decision making. Understandably, an analyst could also be a system engineer. In this section we point out some of the practical concerns that are important to customers in industry and discuss how several of these concerns are dealt with in the process of attempting to validate and calibrate the results of the codes.

As the use of codes, particularly commercial codes, in industry becomes more widespread, industrial customers are requiring that these codes be more responsive to their needs and demonstrate more reliability. The primary concern of the industrial code user is the level of confidence he has in the daily production use of the code, particularly in situations where the validation domain is poorly known (see Fig. 9). The accuracy and reliability demands of industrial customers vary greatly, however, based primarily on the level of physical detail in which the customers are interested. This spectrum of physical complexity can range, for example, from output quantities restricted to a limited number of global response measures, such as, engine or compressor efficiency, drag and lift coefficients, global forces and moments, up to detailed physical characteristics, such as local temperature or plastic strain in selected regions, or to gradients of properties, such as local heat flux or fluid dynamic turbulence quantities. Given this spectrum of possible physical complexity, the validation requirements of industrial customers will vary greatly.

An illustration of how the validation requirements for industrial customers can vary is taken from the ASME benchmark exercise of the transonic compressor rotor R37 from NASA [201]. This exercise was handled as a blind validation computation by a large number of codes. When the results were compiled at the conclusion of the exercise, the scatter of the computational results on global compressor efficiency was on the order of $\pm 2\%$. This small level of scatter was welcomed in the CFD community, but the important observation from the exercise was that different computations showing the same level of accuracy for computed efficiency were due to very different combinations of errors in numerics and physical modeling. Since the global compressor efficiency is obtained by a ratio involving average stagnation pressure and temperature, the same value of efficiency can be obtained by underpredicting both pressure and temperature or by exactly the opposite, overpredicting both quantities. In essence, different numerators and different denominators can produce the same acceptable global efficiency result. This kind of example raises serious and difficult questions for code developers regarding the importance of understanding and responding to a wide range of customer requirements. To some customers, the accuracy of the ratio would be sufficient. To other customers, the accuracy requirements would dictate that both numerator and denominator be correct, not just the ratio of the two quantities. We believe it is just as important to understand *why* a comparison between computation and experiment is good or bad, as it is to obtain good agreement between computation and experiment.

All predictions will inevitably include a wide range of uncertainties, given the complexity of the systems being modeled. Industrial customers must find ways to cope with

these uncertainties, as opposed to the initial expectation of accurately quantifying them. Generally, industrial customers tend to use computational models in more pragmatic ways than do code developers and researchers [202]. The rigorous application of computational models by code developers and researchers is often a luxury not available to industrial customers, who are constrained by the time and costs of industry design procedures. These constraints have consequences on the level of requirements that industry can put on a simulation. For example, industrial simulations tend to limit grid density, grid quality, and iterative convergence bounds; and sometimes the industrial simulations do not estimate the modeling uncertainty by comparing the results from different physical models. Consequently, the goals of numerical error estimation and the validation recommendations discussed previously are clearly out of reach in many industrial environments. Instead, the dialog between code developers and their customers focuses on the critical issue of *managing the uncertainties*. This is a new concept in computational simulation, but it is analogous to risk management in engineering design projects. This concept requires significant research to define rational methodologies, quantitative evaluations, and techniques for integrating the management of uncertainties in the decision-making process.

One way industry is attempting to deal with the severe constraints of cost and schedule affecting numerical error estimation and model uncertainty estimation is to concentrate on the ''deltas'' from computations. That is, the analyst considers only the changes of output quantities from run to run that are due to, for example, changes in geometry or boundary conditions. The basic assumption in this approach is that, for a given level of combined errors, the balance of errors will not vary significantly when relatively small changes in geometrical or input parameters are performed during a design-cycle process. This assumption is reasonable and pragmatic, but there is no assurance that the deltas will be accurate.

To improve the reliability of this approach, industrial customers calibrate their computational results with experimental data that are available for their existing hardware products. For example, industrial customers typically have large quantities of proprietary experimental data for different response measures of their products. As part of their calibration procedure, industrial customers will choose the option for the mathematical model in the code that best reproduces a wide range of their experimental data. When this calibration procedure is used, one of the important requirements is that there be ''no change'' in computational results between old and new versions of the code for the same set of geometrical conditions, boundary conditions, grids, and physical models chosen. This no-change requirement allows an industrial code user to keep the same calibration of parameters for the way in which the code was used. This calibration procedure in no way should be viewed as validation. It is a method of ensuring that all of the possible numerical error sources and all of the physical model uncertainties consistently sum to zero for a given set of response measures. The issue of summation and cancellation of errors and uncertainties is addressed in [12].

A recent benchmark exercise organized by AIAA and NASA to validate lift and drag predictions of a wing-body combination illustrates the concerns of industrial customers for absolute predictive capability in a most striking way [203,204]. This exercise, conducted as a workshop, was set up to assess the level of confidence of CFD codes for force and moment predictions for the DLR-F4 wing-body data set. Thirty-five different CFD calculations were presented using various grids, both structured and unstructured, including some grids recommended to minimize grid dependency on some of the test cases. Although most of the codes were among the most advanced research codes available, providing better results than did some of the commercial codes, the outcome clearly showed the current limitations of CFD. Statistical analysis indicated that the standard deviation of the CFD results, even after exclusion of the computational ''outliers,'' was still 21 drag counts. (Note that 1 drag count is equal to a drag coefficient of 0.0001.) In comparison, wind tunnel experimental data is believed to be accurate to 4 drag counts. The industry objective, however, is 1 drag count; that is, a result on the order of 0.5% of the total drag of a modern transport aircraft. Citing from [203]:

> More experience needs to be gained where CFD is used in conjunction with wind tunnel data on development projects that culminate in a flight vehicle. Then the methods can be ''calibrated'' to a known outcome. Note that experimental methods went through a similar process long ago. Wind tunnel testing is not regarded as ''perfect,'' but it is useful as an engineering tool because its advantages and limitations are well known. CFD needs to go through the same process.

4.3 Commercial software

This section addresses implementation issues in V&V that are relevant to commercial software companies, the commercial software packages they produce, and the customers with whom these companies interact.

Commercial software companies are subjected to severe quality and reliability requirements from their customers. In addition, many of the recommended activities discussed in this paper which, for example, focused attention on modeling and simulation requirements, cannot be practicably carried out by the commercial code company because of the extremely wide range of applications required by its customers. In light of such demands, the following factors are important for these companies to consider as part of their business activities:

1) Both the software company and the customer must understand that it is practically impossible to cover all possible combinations of physical, boundary, and geometrical conditions during the V&V process. As pointed out repeatedly in this paper, a code cannot be validated for all possible physical modeling options available in the code. Hence, both parties, software company and customer, must learn to cope with management of the uncertainties.

2) The software company must put in place and practice a high level of SQA for every stage of the software life cycle. These stages range from a new software module to intermediate releases, to simple bug-fixing patches. Each of these types of code releases will require separate SQA procedures for its quality assessment.
3) A commercial software system is much more than a numerical solver of the physical models. The system also contains the global communication and interaction system between the user, the various modules of the software system, and the computer as accessed through the graphical user interface (GUI). The GUI must act in the following ways: a) as the task manager, allowing the user to direct computer runs to different computer systems on the network and/or launching a series of runs with varying parameters; b) as a preprocessing unit for input geometries provided by computer-aided design (CAD) software; c) as an interface for grid generation tools; d) as input data to establish all of the needed boundary conditions and initial conditions; and e) as the mechanism to transfer data to the postprocessing software.
4) The complete software system has to be compiled and ported to a wide variety of computer platforms (both serial and parallel processor machines), which run under different operating systems, and must be verified for a range of compiler options.

The V&V procedures within a software company environment constitute an ongoing process. With every software release, the company must have automatic SQA test procedures that are set up under the supervision of a software quality group. These procedures must include automatic reporting features for the results of the testing. In particular, the activities related to fixing, tracking, and managing software bugs are a critical part of the global SQA procedures.

Commercial software companies have long recognized the importance of documentation for their products. This documentation has mainly described the use of the GUI, the capabilities of the code, grid generation capabilities, numerical algorithm techniques, and postprocessing capabilities. However, it is our view that V&V activities have been poorly documented, if at all. This claim, of course, is also valid for most software development projects in computational physics. It is necessary to document the content and results of all major V&V activities, even though this practice could present a risk to the software company in relation to its competitors. The old adage of quality engineering applies here as well: If it isn't documented, it didn't happen.

Documentation can be in traditional paper form, or in the form of a computer-based records management system. It is our belief that a computer-based system will provide the best vehicle for two reasons. First, a computer-based system can more easily be updated and accessed, and it can include much more detail. For example, documentation can be provided on how the code was used in the definition, design, and analysis of each validation experiment. Enough information should be included to allow reproduction of the described calculations by others. Additionally, a complete description of each experiment should be given, and all of the experimental data should be easily available for visual and quantitative presentation. Second, a properly designed computer-based system can easily be searched from a wide variety of perspectives, eg, computational, experimental, and engineering project, and can point or link to other documented information associated with the V&V activities that intersects a wide range of perspectives. Internet access greatly facilitates these desirable search and link capabilities, both within an organization's computer network and between organizations that are widely spread geographically but that work together as part of a project team.

One important way that software companies can improve the reliability of their codes is to provide sufficient training and education for their customers. For new users of a commercial code, the training should focus on basic operation of the code and use of the GUI interface. For more experienced users, the training should focus on techniques for sophisticated grid generation, on understanding the limitations of models in the code, and on approaches for dealing with the wide range of uncertainties in real engineering systems. It is our view that large organizations that develop their own codes for internal use sometimes neglect to adequately address user support issues such as training, bug fixing, and consulting on the use of their codes.

4.4 Development of standards

The maturation of V&V depends on concerted efforts by individuals and groups throughout the world to develop standardized terminology, procedures, and tools that can be used by the computational physics community. In this section we call for a further commitment to such efforts, recommend the need to clarify the present terminology on validation, address the contributions that standards organizations can make and are making in this process, and underscore the critical need for benchmark databases for both verification and validation. Current European efforts to construct validation databases are highlighted.

Concerning terminology, we have used the DoD/AIAA definition of validation to mean that validation does *not* include the concept of *adequacy of the computational result for the intended uses* of the model. We argue that this is what the words mean in the definition of validation given in Section 1.2. As we noted, however, our interpretation is narrower than the interpretation that is widely accepted in the DoD community. The DoD community considers validation to be the process of determining the degree to which the model is *adequate* for the application of interest. We argue that there are two significant reasons not to use the meaning ascribed by the DoD community. First, if the DoD interpretation were used, then individual researchers or analysts who are comparing computational and experimental results *could not* claim they were validating a model. The *only* individuals, or group of individuals, that can do this are those who understand the intended application for the modeling and simulation capability and can state that the level of agreement between computation and experiment is adequate for the intended application. Second, the adequacy requirements for

the intended application are typically ill defined. Adequacy requirements are ill defined for practical reasons; for example, adequacy may be defined fairly well at the complete system tier but very poorly for lower tiers of the validation hierarchy. Also, adequacy requirements are ill defined for technical reasons; for example, the analysis of complex systems involves the interaction of many different types of computational models, each with its own adequacy. The computational adequacy for some high-level system response measures is actually a combination of many adequacy measures from each of the contributing models. As a result, one must deal with adequacy requirements at a system level in an iterative and coupled manner by propagating uncertainty through many different types of physics models and multiple tiers of hierarchical complexity. Whether or not one accepts the arguments given here for our interpretation, the issue of interpretation of the meaning of validation needs further discussion and debate by the DoD community, the DOE community, engineering standards-writing committees, and other stakeholders relying on modeling and simulation.

We believe there is an urgent need to develop industry-wide standards for V&V procedures and tools that are applicable to the computational physics community. In our view, the most appropriate organizations for defining these standards are professional engineering societies that have officially recognized standards-writing committees. A complementary approach, one that is appropriate at the worldwide level, is the involvement of the International Organization for Standardization (ISO). We believe there is also an important contribution to be made by national standards organizations, such as the US National Institute of Standards and Technology (NIST), and by similar organizations in the European Union, such as the European Research Community on Flow, Turbulence, And Combustion (ERCOFTAC) [205], and the National Agency for Finite Element Methods and Standards (NAFEMS) [206].

We believe standards organizations are best able to contribute in two areas of V&V: codification of recommended procedures and tools. As discussed earlier in this paper, the AIAA CFD Committee on Standards contributed the first standards document on V&V in 1998 [9]. The document provided a foundation on which to build, but it was primarily focused on background, terminology, and fundamental concepts. New standards documents are needed that go beyond the AIAA Guide in terms of detailed procedures and tools, and that address technical disciplines other than fluid dynamics. For example, a new standards committee was formed in ASME: the Committee on Verification and Validation in Computational Solid Mechanics (Performance Test Codes No. 60). In Europe, a particularly effective activity for the development of V&V procedures has been conducted by ERCOFTAC. ERCOFTAC privately funded an initiative on ''Quality and Trust in Industrial CFD,'' which led to the publication of the comprehensive report by Casey and Wintergerste [28]. This document, although not a standards document, gives general as well as specific practical advice on modeling and numerical techniques for CFD.

Concerning V&V tools, we contend that the key tools that need to be constructed are separate benchmark databases for verification and validation. Verification databases should be composed of all three types of highly accurate solutions, which we will simply refer to as benchmark solutions. *Strong-sense benchmarks* are test problems that are precisely defined and formally documented, typically by professional societies, academic institutions, or nonprofit organizations. It is our view that a strong-sense benchmark is described by the following four factors:

1) An exact, standardized, frozen, and promulgated definition of the benchmark.
2) An exact, standardized, and promulgated statement of the purpose of the benchmark. This statement addresses the benchmark's role and application in a comprehensive test plan for a code, for example.
3) Exact, standardized, frozen, and promulgated requirements for comparison of codes with the benchmark's results.
4) An exact, standardized, frozen, and promulgated definition of acceptance criteria for comparison of codes with the benchmark's results. The criteria can be phrased either in terms of success or failure.

Do strong-sense benchmarks exist? We believe that the answer is ''certainly not in computational physics.'' We believe that standards organizations are the most appropriate organizations to compile and codify strong-sense benchmarks. We note that none of these groups have dealt with the most sensitive issue in our definition of strong-sense benchmarks, namely, the issue of establishing stringent standards for comparison with benchmarks and measuring success or failure. It is very clear to us that establishing standards will be a long and difficult process. The realities of competition among commercially available computational physics codes, competition among companies that rely on computational physics codes, and competition among countries may stifle this endeavor. However, we believe that the maturation of computational physics will suffer if strong-sense benchmarks are not developed.

Validation benchmark databases are also critically needed. In the near term, these databases should be populated using existing high-quality experimental data that meet most, if not all, of the requirements for validation experiments specified earlier in this paper. The construction of experimental databases is in development both in the United States and in Europe. In the United States, for example, the National Project for Applications-oriented Research in CFD (NPARC) initiative has concentrated on validation of internal reacting flows. In Europe, the FLOWNET project is constructing databases for a wide range of industrial applications in support of validation. Closely related to FLOWNET is the QNET-CFD Thematic Network [207]. This network has more than 40 participants from several countries who represent research establishments and many sectors of the industry, including commercial CFD software companies. The QNET-CFD Thematic Network is funded by the European Commission. The objective of the network is to establish a list of so-called Application Challenges, or ACs, and Underlying Flow Re-

gime Challenges. The Application Challenges are specific applications that represent simulations at the subsystem tier shown in Fig. 10. The Underlying Flow Regime Challenges represent simulations similar to the benchmark tier shown in Fig. 10.

The QNET-CFD Thematic Network has completed its second year of activity, with the participating organizations contributing 53 Application Challenges divided over six Thematic Areas aligned with the following industrial sectors: External Aerodynamics (10 ACs); Combustion & Heat Transfer (7 ACs); Chemical & Process, Thermal Hydraulics, and Nuclear Safety (11 ACs); Civil Construction (6 ACs); Environment (7 ACs); and Turbomachinery Internal Flows (12 ACs). The main requirement for each Application Challenge is that the experimental results become available so that CFD validation can be conducted. Even though the QNET-CFD Thematic Network is relatively new, it is our view that it is an exemplary initiative in the construction of validation databases. We believe that industry, government, and academia in the United States should promptly begin to formulate a similar activity.

In the long term, new validation experiments should be funded, and these results should then be entered in the validation database. We believe that identification of the validation experiments should be the responsibility of the application community. Of course, there would probably be little agreement in the application community on the specific experiments to be conducted. Funding for high-priority experiments could be obtained from research establishments, governmental institutions, and even joint ventures between private industrial organizations. The organizational role and facilitation of discussions regarding which experiments should be conducted would be best served by standards organizations. The validation databases, constructed in both the near term and the long term, could either be completely open to the public or accessible only to member organizations.

5 CLOSING REMARKS

Implementation of most of the approaches and procedures recommended here, for V&V computations and experiments, will be neither inexpensive nor easy. Furthermore, some of these approaches may even be technically or economically impractical in particular situations. In addition, some of the approaches and procedures have not been developed satisfactorily for implementation in an engineering environment. With each included step, however, the quality of the V&V processes will be improved, resulting in increased confidence in the modeling and simulation capability. We firmly believe that V&V is a process, *not* a product. We also believe that an effective path forward to improving the V&V processes is to contrast processes that are ideal, possible, and ineffective. The following gives examples of these as they relate to various V&V activities, as viewed from the engineering perspective of developing modeling and simulation capability:

1) **Ideal**: The application domain for the modeling and simulation capability is well understood and carefully defined, and the accuracy requirements are known.
   Possible: Portions of the application domain are understood and some of the accuracy requirements are known.
   Ineffective: The modeling and simulation capability is expected to accurately compute results for every advertised option in the code.
2) **Ideal**: The validation tier hierarchy, analogous to the validation pyramid discussed, has been carefully constructed using a systems engineering approach.
   Possible: The validation hierarchy has been constructed for individual subsystems or components, but the interactions between subsystems have not been studied.
   Ineffective: No validation hierarchy has been constructed and only expedient validation experiments have been identified.
3) **Ideal**: The PIRT has been used to identify and rank the importance of physical processes and interactions of processes for all tiers and faces of the validation pyramid.
   Possible: The PIRT has been used to identify and rank processes on one face of the validation pyramid.
   Ineffective: No PIRT was constructed and only well exercised physical-process options in the code are considered for V&V activities.
4) **Ideal**: Using the results of the PIRT, code verification activities have been defined and prioritized, schedules set, and needed resources allocated.
   Possible: Using the results of the PIRT, some of the code verification activities are defined, but the code development resources or schedule dictate which activities will be conducted.
   Ineffective: No PIRT was constructed and code verification activities are haphazard and only conducted if resources and time become available.
5) **Ideal**: SQA procedures have been defined, implemented, consistently adhered to, and documented.
   Possible: SQA procedures are poorly defined and partially implemented, only adhered to just before a new code release, and informally documented.
   Ineffective: SQA procedures have not been considered or implemented, and are only mentioned if the code customer asks about them.
6) **Ideal**: Using the results of the PIRT, model validation activities, both computational and experimental, have been defined and prioritized, schedules set, and needed resources allocated.
   Possible: The PIRT is used to define and prioritize some model validation activities, but code-application schedules and limited resources require elimination of many validation activities.
   Ineffective: No PIRT was constructed and model validation will only be conducted using existing experimental data.
7) **Ideal**: The code is used to define, design, and analyze all of the validation experiments.
   Possible: The code is applied to define and design some of the validation experiments, but many of the code results are only available after the validation experiment is completed.
   Ineffective: The validation experiments are conducted completely separate from any input from the code development team and code results.
8) **Ideal**: Highly converged numerical solutions, in both space and time, have been obtained for every validation experiment conducted.
   Possible: Quantitative numerical error estimates are computed for most validation experiments, but some complex-system experiments only have qualitative error estimates.
   Ineffective: No quantitative numerical error estimates are available and only single-grid, single-time-step solutions are available.

9) **Ideal**: Validation-experiment data are well characterized, and a large number of experimental realizations are available for estimating random and bias errors.
Possible: The experimental data are moderately characterized, and only two experimental realizations are obtained.
Ineffective: Only one experimental realization is available and no estimates are available for random or bias errors in needed computational quantities or in measured experimental results.
10) **Ideal**: Validation metrics are carefully defined and requirements stated, and clearly connected to modeling and simulation requirements for the application domain.
Possible: Validation metrics are only defined and requirements stated for high-level system responses, and not defined or requirements stated for low-level (ie, unit tier) physical response.
Ineffective: Validation metrics are not defined and only qualitative assessment is to be used for low-level physical responses.
11) **Ideal**: Using statistical data for both computational and experimental results for the validation metrics, one is able to precisely quantify the accuracy of the model for all validation experiments.
Possible: Statistical data are available from the experimental results but computational results are limited to deterministic metrics because of highly constrained budgets and schedules.
Ineffective: Validation metrics are not defined and the ''view-graph norm'' is used for assessment.
12) **Ideal**: Lessons learned from all of the validation activities are clearly formulated, documented, and broadly communicated.
Possible: Lessons learned from some validation experiments are documented, but only those where the validation experiment and the code accuracy was considered a success.
Ineffective: No time or resources are available for determining or documenting lessons learned, and lessons that are learned are verbally and randomly communicated.

Some will argue that the costs of V&V activities can exceed their value added. In some instances they might. However, we remind these skeptics that the costs of V&V must be weighed against the costs of incorrect or improper decisions made based on computational modeling and simulation. Analogous to probabilistic risk assessment activities, the risk is typically defined as the product of the probability of the occurrence of the event and the consequence of the event. For example, suppose an erroneous conclusion is made on the physics of some process described in an article in a research journal. The erroneous result would rightly be viewed as a low-consequence risk. On the other hand, if erroneous conclusions based on modeling and simulation are made on high-consequence events, decision makers could place their constituency at extreme risk. This is exceedingly more true for systems that cannot be tested. For these situations, the only path forward is to *drastically improve* our confidence and understanding in computational simulations.

ACKNOWLEDGMENTS

The authors sincerely thank Dean Dobranich, Robert Paulsen, and Marty Pilch of Sandia National Laboratories and Patrick Roache, private consultant, for reviewing an earlier version of the manuscript and providing many helpful suggestions for improvements. The first author thanks Robert Thomas of Sandia National Labs for his generous support to complete this work. We also thank Rhonda Reinert of Technically Write, Inc for providing extensive editorial assistance during the writing of the manuscript.

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-AL85000.

REFERENCES

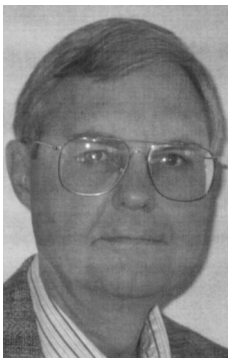
[1] DoD (1994), *DoD directive No 5000.59: Modeling and Simulation (M&S) Management*, Defense Modeling and Simulation Office, Office of the Director of Defense Research and Engineering.
[2] DoD (1996), *Verification, Validation, and Accreditation (VV&A) Recommended Practices Guide*, Defense Modeling and Simulation Office, Office of the Director of Defense Research and Engineering.
[3] Cohen ML, Rolph JE, and Steffey DL (eds) (1998), *Statistics, Testing, and Defense Acquisition: New Approaches and Methodological Improvements*, National Academy Press, Washington DC.
[4] IEEE (1984), *IEEE Standard Dictionary of Electrical and Electronics Terms*, ANSI/IEEE Std 100-1984, New York.
[5] ANS (1987), *Guidelines for the Verification and Validation of Scientific and Engineering Computer Programs for the Nuclear Industry*, Am Nuc Soc, ANSI/ANS-10.4-1987, La Grange Park IL.
[6] IEEE (1991), *IEEE Standard Glossary of Software Engineering Terminology*, IEEE Std 610.12-1990, New York.
[7] ISO (1991), ISO 9000-3: *Quality Management and Quality Assurance Standards-Part 3: Guidelines for the Application of ISO 9001 to the Development, Supply, and Maintenance of Software*, Int Standards Organization, Geneva, Switzerland.
[8] DoD (1996), *DoD Instruction 5000.61: Modeling and Simulation (M&S) Verification, Validation, and Accreditation (VV&A)*, Defense Modeling and Simulation Office, Office of the Director of Defense Research and Engineering.
[9] AIAA (1998), *Guide for the Verification and Validation of Computational Fluid Dynamics Simulations*, American Institute of Aeronautics and Astronautics, AIAA, AIAA-G-077-1998, Reston VA.
[10] Oberkampf WL (1994), A proposed framework for computational fluid dynamics code calibration/validation, American Institute of Aeronautics and Astronautics, AIAA Paper No. 94-2540, 18th AIAA Aerospace Ground Testing Conf, Colorado Springs CO.
[11] Oberkampf WL and Trucano TG (2000), Validation methodology in computational fluid dynamics, Am Inst of Aeronaut and Astronaut, AIAA 2000-2549, Fluids 2000 Conf, Denver CO.
[12] Oberkampf WL and Trucano TG (2002), Verification and validation in computational fluid dynamics, *Prog. Aerosp. Sci.* **38**(3), 209–272.
[13] Schlesinger S (1979), Terminology for model credibility, *Simulation* **32**(3), 103–104.
[14] Roache PJ (1998), *Verification and Validation in Computational Science and Engineering*, Hermosa Publ, Albuquerque NM.
[15] Knepell PL and Arangno DC (1993), *Simulation Validation: A Confidence Assessment Methodology*, 1st Edition, IEEE Comput Soc Press, Washington DC.
[16] Roache PJ (1998), Verification of codes and calculations, *AIAA J.* **36**(5), 696–702.
[17] Lin SJ, Barson SL, and Sindir MM (1992), Development of evaluation criteria and a procedure for assessing predictive capability and code performance, *Advanced Earth-to-Orbit Propulsion Technology Conf*, Marshall Space Flight Center, Huntsville AL.
[18] Marvin JG (1995), Perspective on computational fluid dynamics validation, *AIAA J.* **33**(10), 1778–1787.
[19] Cosner RR (1995), CFD validation requirements for technology transition, Am Inst of Aeronaut and Astronaut, AIAA Paper No. 95-2227, 26th AIAA Fluid Dynamics Conf, San Diego CA.
[20] Sindir MM, Barson SL, Chan DC, and Lin WH (1996), On the development and demonstration of a code validation process for industrial applications, Am Inst of Aeronaut and Astronaut, AIAA Paper No. 96-2032, 27th AIAA Fluid Dynamics Conf, New Orleans LA.
[21] Sindir MM and Lynch ED (1997), Overview of the state-of-practice of computational fluid dynamics in advanced propulsion system de-

sign, Am. Inst. Aeronaut. Astronaut, AIAA Paper No. 97-2124, 28th AIAA Fluid Dynamics Conf, Snowmass CO.
[22] Kleijner JPC (1995), Verification and validation of simulation models, *Eur. J. Oper. Res.* **82**, 145–162.
[23] Balci O (1997), Principles of simulation of model validation, verification, and testing, *Trans. Soc. Comput. Simul. Int.* **14**, 3–12.
[24] Kleindorfer GB, O'Neill L, and Ganeshan R (1998), Validation in simulation: various positions in the philosophy of science, *Manage. Sci.* **44**(8), 1087–1099.
[25] Murray-Smith DJ (1998), Methods for the external validation of continuous systems simulation models: A review, *Mathematical and Computer Modelling of Dynamics Systems* **4**, 5–31.
[26] Robinson S (1999), Simulation verification, validation, and confidence: A tutorial, *Trans. Soc. Comput. Simul. Int.* **16**, 63–69.
[27] Zeigler BP, Praehofer H, and Kim TG (2000), *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems, 2nd Edition*, Academic Press, San Diego CA.
[28] Casey M and Wintergerste T (eds) (2000), *ERCOFTAC Special Interest Group on Quality and Trust in Industrial CFD: Best Practices Guidelines*, European Research Community on Flow, Turbulence, and Combustion.
[29] Pilch M, Trucano TG, Moya JL, Froehlich GK, Hodges AL, and Peercy DE (2001), *Guidelines for Sandia ASCI Verification and Validation Plans-Content and Format: Version 2*, Sandia National Laboratories, SAND2000-3101, Albuquerque NM.
[30] Trucano TG, Pilch M, and Oberkampf WL (2002), General concepts for experimental validation of ASCI code applications, Sandia National Laboratories, SAND2002-0341, Albuquerque NM.
[31] Easterling RG (2001), Measuring the predictive capability of computational models: principles and methods, issues and illustrations, Sandia National Laboratories, SAND2001-0243, Albuquerque NM.
[32] Chiles J-P and Delfiner P (1999), *Geostatistics: Modeling Spatial Uncertainty*, John Wiley, New York.
[33] Oberkampf WL, Diegert KV, Alvin KF, and Rutherford BM (1998), Variability, uncertainty, and error in computational simulations, ASME, ASME-HTD-Vol 357-2, AIAA/ASME Joint Thermophysics and Heat Transfer Conf, Albuquerque NM.
[34] Oberkampf WL, DeLand SM, Rutherford BM, Diegert KV, and Alvin KF (2000), Estimation of total uncertainty in computational simulation, Sandia National Laboratories, SAND2000-0824, Albuquerque NM.
[35] Oberkampf WL, DeLand SM, Rutherford BM, Diegert KV, and Alvin KF (2002), Error and uncertainty in modeling and simulation, *Reliability Eng. Sys. Safety* **75**(3), 333–357.
[36] Coleman HW and Steele Jr WG (1999), *Experimentation and Uncertainty Analysis for Engineers*, 2nd Edition, John Wiley, New York.
[37] Hora SC (1996), Aleatory and epistemic uncertainty in probability elicitation with an example from hazardous waste management, *Reliability Eng. Sys. Safety* **54**, 217–223.
[38] Ferson S, and Ginzburg LR (1996), Different methods are needed to propagate ignorance and variability, *Reliability Eng. Sys. Safety* **54**, 133–144.
[39] Helton JC (1997), Uncertainty and sensitivity analysis in the presence of stochastic and subjective uncertainty, *J. Stat. Comput. Simul.* **57**, 3–76.
[40] Paté-Cornell ME (1996), Uncertainties in risk analysis: Six levels of treatment, *Reliability Eng. Sys. Safety* **54**, 95–111.
[41] Cullen AC and Frey HC (1999), *Probabilistic Techniques in Exposure Assessment: A Handbook for Dealing with Variability and Uncertainty in Models and Inputs*, Plenum Press, New York.
[42] Frank MV (1999), Treatment of uncertainties in space nuclear risk assessment with examples from Cassini mission applications, *Reliability Eng. Sys. Safety* **66**, 203–221.
[43] Smithson M (1989), *Ignorance and Uncertainty: Emerging Paradigms*, Springer-Verlag, New York.
[44] Almond RG (1995), *Graphical Belief Modeling, 1st Edition*, Chapman & Hall, London.
[45] Kohlas J and Monney P-A (1995), *A Mathematical Theory of Hints-An Approach to the Dempster-Shafer Theory of Evidence*, Springer, Berlin.
[46] Klir GJ and Wierman MJ (1998), *Uncertainty-based information: elements of generalized information theory*, Physica-Verlag, Heidelberg.
[47] Dubois D and Prade H (eds) (2000), *Fundamentals of Fuzzy Sets*, Kluwer Academic Publ, Boston MA.
[48] Kramosil I (2001), *Probabilistic Analysis of Belief Functions*, Kluwer, New York.
[49] NRC (1990), Severe accident risks: An assessment for five US nuclear power plants, US Nuclear Regulatory Commission, Office of Nuclear Regulatory Research, Div of Systems Research, NUREG-1150, Washington DC.
[50] Modarres M (1993), *What Every Engineer Should Know about Reliability and Risk Analysis*, Marcel Dekker, New York.
[51] Kafka P (1994), Important issues using PSA technology for design of new systems and plants, *Reliability Eng. Sys. Safety* **45**(1–2), 205–213.
[52] Kumamoto H and Henley EJ (1996), *Probabilistic Risk Assessment and Management for Engineers and Scientists*, 2nd Edition, IEEE Press, New York.
[53] Helton JC, Anderson DR, Baker BL, Bean JE, Berglund JW, Beyeler W, Economy K, Garner JW, Hora SC, Iuzzolino HJ, Knupp P, Marietta MG, Rath J, Rechard RP, Roache PJ, *et al* (1996), Uncertainty and sensitivity analysis results obtained in the 1992 performance assessment for the Waste Isolation Pilot Plant, *Reliability Eng. Sys. Safety* **51**(1), 53–100.
[54] Paté-Cornell ME (1999), Conditional uncertainty analysis and implications for decision making: The case of WIPP, *Risk Anal* **19**(5), 1003–1016.
[55] Helton JC, Anderson DR, Basabilvazo G, Jow H-N, and Marietta MG (2000), Conceptual structure of the 1996 performance assessment for the Waste Isolation Pilot Plant, *Reliability Eng. Sys. Safety* **69**(1–3), 151–165.
[56] Mohanty S, Codell RB, Ahn TM, and Cragnolino GA (2000), An approach to the assessment of high-level radioactive waste containment II: Radionuclide releases from an engineered barrier system, *Nucl. Eng. Des.* **201**, 307–325.
[57] Boyack BE, Catton I, Duffey RB, Griffith P, Katsma KR, Lellouche GS, Levy S, Rohatgi US, Wilson GE, Wulff W, and Zuber N (1990), Quantifying reactor safety margins, Part 1: An overview of the code scaling, applicability, and uncertainty evaluation methodology, *Nucl. Eng. Des.* **119**, 1–15.
[58] Wilson GE, Boyack BE, Catton I, Duffey RB, Griffith P, Katsma KR, Lellouche GS, Levy S, Rohatgi US, Wulff W, and Zuber N (1990), Quantifying reactor safety margins, Part 2: Characterization of important contributors to uncertainty, *Nucl. Eng. Des.* **119**, 17–31.
[59] Wulff W, Boyack BE, Catton I, Duffey RB, Griffith P, Katsma KR, Lellouche GS, Levy S, Rohatgi US, Wilson GE, and Zuber N (1990), Quantifying reactor safety margins, Part 3: Assessment and ranging of parameters, *Nucl. Eng. Des.* **119**, 33–65.
[60] Wilson GE, and Boyack BE (1998), The role of the PIRT in experiments, code development, and code applications associated with reactor safety assessment, *Nucl. Eng. Des.* **186**, 23–37.
[61] Zuber N, Wilson GE, Ishii M, Wulff W, Boyack BE, Dukler AE, Griffith P, Healzer JM, Henry RE, Lehner JR, Levy S, and Moody FJ (1998), An integrated structure and scaling methodology for severe accident technical issue resolution: Development of methodology, *Nucl. Eng. Des.* **186**(1–2), 1–21.
[62] Roache PJ (2002), Code verification by the method of manufactured solutions, *ASME J. Fluids Eng.* **114**(1), 4–10.
[63] Peercy DE (2000), personal communication.
[64] Hatton L (1997), The T experiments: Errors in scientific software, *IEEE Comput. Sci. Eng.* **4**(2), 27–38.
[65] Stevenson DE (1999), A critical look at quality in large-scale simulations, *Comput. Sci. Eng.* **1**(3), 53–63.
[66] Gustafson J (1998), Computational verifiability and feasibility of the ASCI program, *IEEE Comput. Sci. Eng.* **5**(1), 36–45.
[67] Johnson DM (1996), A review of fault management techniques used in safety-critical avionic systems, *Prog. Aerosp. Sci.* **32**(5), 415–431.
[68] FAA (1988), System design analysis, Federal Aviation Administration, Advisory Circular 25.1309-1A, Washington DC.
[69] NIST (2002), The economic impacts of inadequate infrastructure for software testing, National Institute of Standards & Technology, Planning Report 02-3, Washington DC.
[70] Shih TM (1985), A procedure to debug computer programs, *Int. J. Numer. Methods Eng.* **21**(6), 1027–1037.
[71] Salari K and Knupp P (2000), Code verification by the method of manufactured solutions, Sandia National Laboratories, SAND2000-1444, Albuquerque NM.
[72] Steinberg S, and Roache PJ (1985), Symbolic manipulation and computational fluid dynamics, *J. Comput. Phys.* **57**(2), 251–284.
[73] Knupp P and Salari K (2002), *Verification of Computer Codes in Computational Science and Engineering*, Chapman & Hall/CRC, Boca Raton FL.
[74] Keller HB (1969), Accurate difference methods for linear ordinary differential systems subject to linear constraints, *SIAM (Soc. Ind. Appl. Math.) J. Numer. Anal.* **6**, 8–30.
[75] Srivastava BN, Werle MJ, and Davis RT (1979), A finite difference

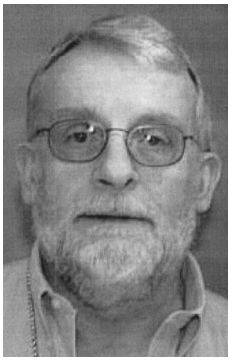
technique involving discontinuous derivatives, *Comput. Fluids* **7**(1), 69–74.
[76] Blottner FG (1982), Influence of boundary approximations and conditions on finite difference solutions, *J. Comput. Phys.* **48**(2), 246–269.
[77] Turkel E (1986), Accuracy of schemes with nonuniform meshes for compressible fluid-flows, *Appl. Numer. Math.* **2**(6), 529–550.
[78] Ferziger JH and Peric M (1996), *Computational Methods for Fluid Dynamics*, Springer-Verlag, New York.
[79] Axelsson O (1996), *Iterative Solution Methods*, Cambridge Univ Press, Cambridge, UK.
[80] Carpenter MH and Casper JH (1999), Accuracy of shock capturing in two spatial dimensions, *AIAA J.* **37**(9), 1072–1079.
[81] Roy CJ, McWherter-Payne MA, and Oberkampf WL (2000), Verification and validation for laminar hypersonic flowfields, Am Inst of Aeronaut and Astronaut, AIAA2000-2550, Fluids 2000 Conf, Denver, CO.
[82] Botella O and Peyret R (2001), Computing singular solutions of the Navier-Stokes equations with the Chebyshev-Collocation method, *Int. J. Numer. Methods Fluids* **36**(2), 125–163.
[83] Roy CJ and Blottner FB (2001), Assessment of one- and two-equation turbulence models for hypersonic flows, *J. Spacecr. Rockets* **38**(5), 699–710.
[84] Diskin B and Thomas JL (2002), Analysis of boundary conditions for factorizable discretizations of the euler equations, NASA/ICASE, NASA/CR-2002-211648, Hampton VA.
[85] Haworth DC, Tahry SHE, and Huebler MS (1993), A global approach to error estimation and physical diagnostics in multidimensional computational fluid dynamics, *Int. J. Numer. Methods Fluids* **17**(1), 75–97.
[86] Singhal AK (1998), Key elements of verification and validation of CFD software, American Institute of Aeronautics and Astronautics, AIAA 98-2639, 29th AIAA Fluid Dynamics Conf, Albuquerque NM.
[87] Wallace DR, Ippolito LM, and Cuthill BB (1996), Reference information for the software verification and validation process, Rept 500-234.
[88] Beizer B (1990), *Software Testing Techniques*, Van Nostrand Reinhold, New York.
[89] Kaner C, Falk J, and Nguyen HQ (1999), *Testing Computer Software*, 2nd Edition, John Wiley, New York.
[90] PSI (1994), *Purecoverage: User's guide*, Pure Software, Inc., Sunnyvale CA.
[91] NRC (1996), *Statistical Software Engineering*, Natl Res Council, Natl Academy Press, Washington DC.
[92] Bowen JP and Hinchey MG (1995), *Applications of Formal Methods*, Prentice-Hall, Englewood Cliffs NJ.
[93] Partsch HA (1990), *Specification and Transformation of Programs*, Springer-Verlag, New York.
[94] Rushby J (1993), Formal methods and the certification of critical systems, Computer Science Lab, SRI Int, SRI-CSL-93-7, Menlo Park CA.
[95] Bowen JP and Hinchey MG (1995), 10-commandments of formal methods, *Computer* **28**(4), 56–63.
[96] Clark Jr EM, Grumberg O, and Peled D (1999), *Model Checking*, MIT Press, Cambridge MA.
[97] Kurshan RP (2000), Program verification, *Not. Am. Math. Soc.*, **47**(5), 534–545.
[98] Fairley RE (1985), *Software Engineering Concepts*, McGraw-Hill, New York.
[99] Baber R (1987), *The Spine of Software: Designing Provably Correct Software: Theory and Practice*, John Wiley, New York.
[100] DeMillo RA, McCracken WM, Martin RJ, and Passafiume JF (1987), *Software Testing and Evaluation*, Benjamin/Cummings, Menlo Park CA.
[101] Dahl O (1992), *Verifiable Programming*, Prentice-Hall, Englewood Cliffs NJ.
[102] Dyer M (1992), *The Cleanroom Approach to Quality Software Development*, John Wiley, New York.
[103] Lewis RO (1992), *Independent Verification and Validation*, 1st Edition, John Wiley, New York.
[104] Rook P (1990), *Software Reliability Handbook*, Elsevier Science Publ, New York.
[105] Paulk MC, Weber CV, Curtis B, and Chrissis MB (eds) (1995), *The Capability Maturity, Model: Guidelines for Improving the Software Process*, Addison-Wesley, Reading, MA.
[106] Jones C (1997), *Applied Software Measurement*, 2nd Edition, McGraw-Hill, New York.
[107] Persse JR (2001), *Implementing the Capability Maturity Model*, John Wiley, New York.
[108] Hirsch C (1988), *Numerical Computation of Internal and External Flows: Vol 1: Fundamentals of Numerical Discretization*, John Wiley, New York.
[109] Hirsch C (1990), *Numerical Computation of Internal and External Flows: Vol 2: Computational Methods for Inviscid and Viscous Flows*, John Wiley, New York.
[110] Oden JT (1993), Error estimation and control in computational fluid dynamics, *The Mathematics of Finite Elements and Applications*, JR Whiteman (ed), John Wiley, New York, 1–23.
[111] Morton KW (1996), *Numerical Solution of Convection-Diffusion Problems*, CRC Press, Boca Raton FL.
[112] Laney CB (1998), *Computational Gasdynamics*, Cambridge Univ Press, Cambridge, UK.
[113] Ainsworth M and Oden JT (2000), *A Posteriori Error Estimation in Finite Element Analysis*, John Wiley, New York.
[114] Babuska I and Strouboulis T (2001), *The Finite Element Method and its Reliability*, Oxford Univ Press, Oxford, UK.
[115] Roache PJ (1990), Need for control of numerical accuracy, *J. Spacecr. Rockets* **27**(2), 98–102.
[116] Roache PJ (1994), Perspective: A method for uniform reporting of grid refinement studies, *ASME J. Fluids Eng.* **116**, 405–413.
[117] Roache PJ (1997), Quantification of uncertainty in computational fluid dynamics, *Annu. Rev. Fluid Mech.*, 126–160.
[118] Roy CJ (2001), Grid convergence error analysis for mixed-order numerical schemes, Am Inst of Aeronaut and Astronaut, AIAA -2001-2606, AIAA Fluid Dynamics Conf, Anaheim CA.
[119] Eca L and Hoekstra M (2002), An evaluation of verification procedures for CFD applications, *Proc of 24th Symp on Naval Hydrodynamics*, Fukuoka, Japan.
[120] Cadafalch J, Perez-Segarra CC, Consul R, and Oliva A (2002), Verification of finite volume computations on steady state fluid flow and heat transfer, *ASME J. Fluids Eng.*, **124**(1), 11–21.
[121] Chen C-FA, Lotz RD, and Thompson BE (2002), Assessment of numerical uncertainty around shocks and corners on blunt trailing-edge supercritical airfoils, *Comput. Fluids* **31**(1), 25–40.
[122] Venditti DA, and Darmofal DL (2000), Adjoint error estimation and grid adaptation for functional outputs: Application to quasi-one-dimensional flow, *J. Comput. Phys.* **164**(1), 204–227.
[123] Jameson A, and Martinelli L (1998), Mesh refinement and modeling errors in flow simulation, *AIAA J.* **36**(5), 676–686.
[124] Barth TJ and Larson MG (2002), A posteriori error estimates for higher order Godunov finite volume methods on unstructured meshes, Natl Aeronaut and Space Administration, NASA-TR-NAS-02-001.
[125] Estep DJ, Larson MG, and Williams RD (2000), Estimating the error of numerical solutions of systems of reaction-diffusion equations, *Mem. Am. Math. Soc.* **146**(696), 1–109.
[126] Babuska I, Ihlenburg F, Strouboulis T, and Gangaraj SK (1997), A posteriori error estimation for finite element solutions of Helmholtz' equation, Part II: Estimation of the pollution error, *Int. J. Numer. Methods Eng.* **40**, 3883–3900.
[127] Babuska I and Oh H-S (1987), Pollution problem of the p- and h-p versions of the finite element method, *Commun. Appl. Numer. Methods* **3**, 553–561.
[128] Babuska I, Strouboulis T, Upadhyay CS, and Gangaraj SK (1995), A posteriori estimation and adaptive control of the pollution error in the h-version of the finite element method, *Int. J. Numer. Methods Eng.* **38**, 4207–4235.
[129] Oden JT, Feng Y, and Prudhomme S (1998), Local and pollution error estimation for Stokesian flow, *Int. J. Numer. Methods Fluids* **27**, 33–39.
[130] Zhang XD, Pelletier D, Trepanier JY, and Camarero R (2000), Verification of error estimators for the euler equations, Am Inst of Aeronaut and Astronau, AIAA-2000-1001, 38th AIAA Aerospace Sciences Meeting, Reno NV.
[131] Chorin AJ, Kast AP, and Kupferman R (1998), On the prediction of large-scale dynamics using unresolved computations, Lawrence Berkeley National Lab, LBNL-42283, Berkeley CA.
[132] Chorin AJ, Kast AP, and Kupferman R (1998), Optimal prediction of underresolved dynamics, *Proc. Natl. Acad. Sci. U.S.A.* **95**, 4094–4098.
[133] Chorin AJ, Kast AP, and Kupferman R (1999), Unresolved computation and optimal prediction, *Commun. Pure Appl. Math.* **52**, 1231–1254.
[134] Glimm J, Hou S, Kim H, Sharp DH, and Ye K (1999), A probability model for errors in the numerical solutions of a partial differential equation, Los Alamos Natl Lab, LAUR-99-5352, Los Alamos NM.
[135] Glimm J and Sharp DH (1997), Stochastic methods for the prediction of complex multiscale phenomena, Los Alamos Natl Lab, LAUR-97-3748, Los Alamos NM.

[136] McMillan C (1996), personal communication.
[137] Younger SM (1997), personal communication.
[138] Cox DR (1958), *Planning of Experiments*, John Wiley, New York.
[139] Dean A and Voss D (1999), *Design and Analysis of Experiments*, Springer-Verlag, New York.
[140] Gunter BH (1993), How statistical design concepts can improve experimentation in the physical sciences, *Comput. Phys.* **7**(3), 262–272.
[141] Law AM and Kelton WD (1991), *Simulation Modeling and Analysis, 2nd Edition*, McGraw-Hill, New York.
[142] Bossel H (1994), *Modeling and Simulation, 1st Edition*, AK Peters, Wellesley MA.
[143] Haimes YY (1998), *Risk Modeling, Assessment, and Management*, John Wiley, New York.
[144] Oberkampf WL and Aeschliman DP (1992), Joint computational/experimental aerodynamics research on a hypersonic vehicle: Part 1, Experimental results, *AIAA J.* **30**(8), 2000–2009.
[145] Walker MA, and Oberkampf WL (1992), Joint computational/experimental aerodynamics research on a hypersonic vehicle: Part 2, Computational results, *AIAA J.* **30**(8), 2010–2016.
[146] Oberkampf WL, Aeschliman DP, Tate RE, and Henfling JF (1993), Experimental aerodynamics research on a hypersonic vehicle, Sandia Natl Labs, SAND92-1411, Albuquerque NM.
[147] Aeschliman DP, Oberkampf WL, and Blottner FG (1995), A proposed methodology for CFD code verification, calibration, and validation, ICIASF, Paper 95-CH3482-7, 16th Int Congress on Instrumentation for Aerospace Simulation Facilities, Dayton OH.
[148] Oberkampf WL, Aeschliman DP, Henfling JF, and Larson DE (1995), Surface pressure measurements for CFD code validation in hypersonic flow, Am Inst of Aeronaut and Astronaut, AIAA Paper No 95-2273, 26th AIAA Fluid Dynamics Conf, San Diego CA.
[149] Aeschliman DP and Oberkampf WL (1998), Experimental methodology for computational fluid dynamics code validation, *AIAA J.* **36**(5), 733–741.
[150] Oberkampf WL and Blottner FG (1998), Issues in computational fluid dynamics code verification and validation, *AIAA J.* **36**(5), 687–695.
[151] Gamerman D (1997), *Markov Chain Monte Carlo*, Chapman & Hall, London.
[152] Kleijnen JPC (1987), *Statistical Tools for Simulation Practitioners, 1st Edition*, Marcel Dekker, New York.
[153] LeGore T (1990), Predictive software validation methodology for use with experiments having limited replicability, *Benchmark Test Cases for Computational Fluid Dynamics*, I Celik and CJ Freitas (eds), ASME, New York, 21–27.
[154] Helton JC (1993), Uncertainty and sensitivity analysis techniques for use in performance assessment for radioactive waste disposal, *Reliability Eng. Sys. Safety* **42**(2–3), 327–367.
[155] Helton JC (1999), Uncertainty and sensitivity analysis in performance assessment for the Waste Isolation Pilot Plant, *Comput. Phys. Commun.* **117**(1–2), 156–180.
[156] Stockman CT, Garner JW, Helton JC, Johnson JD, Shinta A, and Smith LN (2000), Radionuclide transport in the vicinity of the repository and associated complementary cumulative distribution functions in the 1996 performance assessment for the Waste Isolation Pilot Plant, *Reliability Eng. Sys. Safety* **69**(1–3), 369–396.
[157] Melchers RE (1999), *Structural Reliability Analysis and Prediction*, John Wiley, New York.
[158] Ditlevsen O and Madsen HO (1996), *Structural Reliability Methods*, John Wiley.
[159] Haldar A and Mahadevan S (2000), *Probability, Reliability, and Statistical Methods in Engineering Design*, John Wiley, New York.
[160] Hamill TM and Wilks DS (1995), A probabilistic forecast contest and the difficulty in assessing short-range forecast uncertainty, *Weather and Forecasting* **10**(3), 620–631.
[161] Du J, Mullen SL, and Sanders F (1997), Short-range ensemble forecasting of quantitative precipitation, *Mon. Weather Rev.* **125**(10), 2427–2459.
[162] Alapaty K, Raman S, and Niyogi DS (1997), Uncertainty in the specification of surface characteristics: A study of prediction errors in the boundary layer, *Boundary-Layer Meteorol.* **82**(3), 473–500.
[163] Chlond A and Wolkau A (2000), Large-eddy simulation of a nocturnal stratocumulus-topped marine atmospheric boundary layer: An uncertainty analysis, *Boundary-Layer Meteorol.* **95**(1), 31–55.
[164] Palmer TN (2000), Predicting uncertainty in forecasts of weather and climate, *Rep. Prog. Phys.* **63**, 71–116.
[165] Sanders F, Mullen SL, and Baumhefner DP (2000), Ensemble simulations of explosive cyclogenesis at ranges of 2–5 days, *Mon. Weather Rev.* **128**(8/Pt 2), 2920–2934.
[166] Glimm J and Sharp D (1998), Stochastic methods for the prediction of complex multiscale phenomena, *Q. Appl. Math.* **56**(4), 741–765.
[167] Earman J (1992), *Bayes or Bust?*, MIT Press, Cambridge MA.
[168] Gelman AB, Carlin JS, Stern HS, and Rubin DB (1995), *Bayesian Data Analysis*, Chapman & Hall, London.
[169] French S and Smith JQ (eds) (1997), *The Practice of Bayesian Analysis*, Hodder Arnold, London.
[170] De Volder B, Glimm J, Grove JW, Kang Y, Lee Y, Pao K, Sharp DH, and Ye K (2002), Uncertainty quantification for multiscale simulations, *ASME J. Fluids Eng.* **124**(1), 29–41.
[171] Gass SI (1993), Model accreditation: A rationale and process for determining a numerical rating, *Eur. J. Oper. Res.* **66**, 250–258.
[172] Lee LH and Poolla K (1994), *Statistical Validation for Uncertainty Models*, Springer-Verlag, Lecture Notes in Control and Information Sciences, Vol 202, *Feedback Control, Complexity, and Identification: A festschrift for Professor George Zames*, Montreal, Canada.
[173] Lee LH and Poolla K (1996), On statistical model validation, *ASME J. Dyn. Syst., Meas., Control* **118**, 226–236.
[174] Draper D (1995), Assessment and propagation of model uncertainty, *J. R. Stat. Soc. Ser. B. Methodol.* **57**(1), 45–97.
[175] Kleijnen JPC (1995), Case-study: Statistical validation of simulation models, *Eur. J. Oper. Res.* **87**, 21–34.
[176] Laskey KB (1996), Model uncertainty: Theory and practical implications, *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* **26**(3), 340–348.
[177] Coleman HW and Stern F (1997), Uncertainties and CFD code validation, *ASME J. Fluids Eng.* **119**, 795–803.
[178] Hills RG and Trucano TG (1999), Statistical validation of engineering and scientific models: Background, Sandia Natl Labs, SAND99-1256, Albuquerque NM.
[179] Hanson KM (1999), A framework for assessing uncertainties in simulation predictions, *Physica D* **133**, 179–188.
[180] Hills RG and Trucano TG (2002), Statistical validation of engineering and scientific models: A maximum likelihood based metric, Sandia Natl Labs, SAND2001-1783, Albuquerque NM.
[181] Hills RG and Trucano TG (2001), Statistical validation of engineering and scientific models with application to CTH, Sandia Natl Labs, SAND2001-0312, Albuquerque NM.
[182] Trucano TG, Easterling RG, Dowding KJ, Paez TL, Urbina A, Romero VJ, Rutherford RM, and Hills RG (2001), Description of the Sandia validation metrics project, Sandia Natl Labs, SAND2001-1339, Albuquerque NM.
[183] Dowding K (2001), Quantitative validation of mathematical models, *ASME Int Mech Eng Congress*, New York.
[184] Easterling RG (2001), Quantifying the uncertainty of computational predictions, Sandia Natl Labs, SAND2001-0919C, Albuquerque NM.
[185] Paez T and Urbina A (2001), Validation of structural dynamics models via hypothesis testing, Society of Experimental Mechanics Annual Conf, Portland, OR.
[186] Urbina A and Paez TL (2001), Statistical validation of structural dynamics models, Institute of Environmental Sciences and Technology, Annual Technical Meeting and Exposition of the Inst of Environmental Sciences and Technology, Phoenix AZ.
[187] Paez TL and Urbina A (2002), Validation of mathematical models of complex structural dynamics systems, International Institute of Acoustics and Vibration, *Proc of 9th Int Congress on Sound and Vibration*, Orlando FL.
[188] Wilson RV, Stern F, Coleman HW, and Paterson EG (2001), Comprehensive approach to verification and validation of CFD simulations, Part 2: Application for RANS simulation of a cargo/container ship, *ASME J. Fluids Eng.* **123**(4), 803–810.
[189] Stern F, Wilson RV, Coleman HW, and Paterson EG (2001), Comprehensive approach to verification and validation of CFD simulations, Part 1: Methodology and procedures, *ASME J. Fluids Eng.* **123**(4), 793–802.
[190] Denker M and Woyczynski WA (1998), *Introductory Statistics and Random Phenomena*, Birkhauser, Boston.
[191] Brandt S (1999), *Data Analysis: Statistical and Computational Methods for Scientists and Engineers*, 3rd Edition, Springer, New York.
[192] Singpurwalla ND and Wilson SP (1999), *Statistical Methods in Software Engineering: Reliability and Risk*, Springer-Verlag, Berlin.
[193] Yee HC, and Sweby PK (1998), Aspects of numerical uncertainties in time marching to steady-state numerical solutions, *AIAA J.* **36**(5), 712–724.
[194] Glimm J, Hou SL, Kim HJ, Lee Y, Sharp DH, Ye K, and Zou WS (2001), Risk management for petroleum reservoir production: A simulation-based study of prediction, *Comput. Geosci.* **5**(3), 173–197.
[195] Hills RG and Leslie IH (2003), Statistical validation of engineering and scientific models: Validation experiments to application, Sandia Natl Labs, SAND2003-0706, Albuquerque NM.
[196] Zhang R, and Mahadevan S (2003), Bayesian methodology for reli-

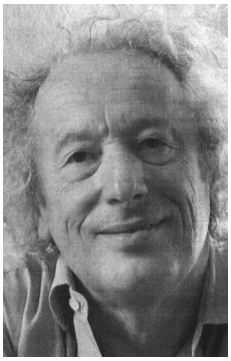
ability model acceptance, *Reliability Eng. Sys. Safety* **80**(1), 95–103.
[197] Campbell K (2002), A brief survey of statistical model calibration ideas, Los Alamos Natl Lab, LA-UR-02-3157, Los Alamos NM.
[198] Hasselman TK (2001), Quantification of uncertainty in structural dynamic models, *ASCE, J. Aerosp. Eng.* **14**(4), 158–165.
[199] Hasselman TK, Wathugala GW, and Crawford J (2002), A hierarchical approach for model validation and uncertainty quantification, Vienna Univ of Technology, Austria, http://wccm.tuwien.ac.at, Fifth World Congress on Computational Mechanics, Vienna, Austria.
[200] Logan RW and Nitta CK (2002), Verification & validation (V&V) methodology and quantitative reliability at confidence (QRC): Basis for an investment strategy, Lawrence Livermore Natl Lab, UCRL-ID-150874, Livermore CA.
[201] Denton JD (1996), Lessons from Rotor 37, *Proc of 3rd Int Symp on Experimental and Computational Aerothermodynamics of Internal Flows*, Beijing, China.
[202] Hutton AG and Casey MV (2001), Quality and trust in industrial CFD—A European perspective, Am Inst of Aeronau and Astronaut, AIAA2001-0656, 39th AIAA Aerospace Sciences Meeting, Reno NV.
[203] Levy DW, Zickuhr T, Vassberg J, Agrawal S, Wahls RW, Pirzadeh S, and Hemsch MJ (2002), Summary of data from the 1st AIAA CFD Drag Prediction Workshop, Am Inst of Aeronaut and Astronaut, AIAA-2002-0841, 40th AIAA Aerospace Sciences Meeting and Exhibit, Reno NV.
[204] Hemsch M (2002), Statistical analysis of CFD solutions from the Drag Prediction Workshop, Am Inst of Aeronaut and Astronaut, AIAA-2002-0842 40th AIAA Aerospace Sciences Meeting and Exhibit, Reno NV.
[205] ERCOFTAC (2000), *Portal to Fluid Dynamics Database Resources*, European Research Community on Flow, Turbulence, and Combustion, http://ercoftac.mech.surrey.ac.uk.
[206] NAFEMS (2000), *CFD Working Group*, International Association for the Engineering Analysis Community, www.NAFEMS.org.
[207] QNET-CFD (2001), *Thematic Network on Quality and Trust for the Industrial Applications of CFD*, www.qnet-cfd.net.



William L Oberkampf *is a Distinguished Member of the Technical Staff at Sandia National Laboratories, Albuquerque, NM. In 1970, after completing his PhD at the University of Notre Dame, he joined the faculty as Assistant Professor in the Mechanical Engineering Department at the University of Texas at Austin. He taught and conducted research in the areas of fluid dynamics, gas dynamics, and computational methods for partial differential equations and in 1976 he was promoted to Associate Professor. In 1979, he joined Sandia and has worked in both staff and management positions in research and development. His work at Sandia has spanned the areas of computational fluid dynamics, hypersonics, laminar and turbulent separated flows, experimental aerodynamics and wind tunnel testing, cavitating flow, and reacting flow. During the last 15 years, Oberkampf has emphasized research in uncertainty estimation in computational mechanics and the development of methodologies for verification and validation of computational simulations. He has 40 journal publications, 93 conference papers and technical reports, 28 special invited lectures, and has taught 8 short courses. He is a member of ASME.*



Timothy G Trucano *is currently a Distinguished Member of the Technical Staff at Sandia National Laboratories in the Optimization and Uncertainty Estimation Department. He received his PhD in mathematical physics from the University of New Mexico. Since 1980, when he started work at Sandia, his work has focused on research, development, and applications of computational shock wave physics. He has worked on problems of equation of state and constitutive modeling, hypervelocity impact phenomena, nuclear weapons effects, and radiation-hydrodynamics in high energy density physics applications. Beginning in 1995, his work began to focus on issues of computational model verification and validation, as well as the application of uncertainty quantification in computational physics. Since 1997, Trucano has worked on technical and programmatic challenges for the Accelerated Strategic Computing Initiative Verification and Validation Program at Sandia. Currently (beginning in Spring of 2002) he is working in the Simulation Enhanced Product Realization systems analysis group at Sandia on problems arising in the interfacing of modeling and simulation with the Sandia nuclear weapons program.*



Charles Hirsch *is currently the Head Professor of the Department of Fluid Mechanics at the VRIJE Universiteit Brussel. In addition, he is a part-time Professor at the Université Libre de Bruxelles (ULB) as well as President of NUMECA International. He received his PhD in Aerodynamics from the ULB in 1969. His practical experience and research activities include: internal flows in turbomachinery, computational methods for internal and external flows, turbulence and transition, turbulence modeling wind effects on buildings and structures, and development and application of wind energy systems. Hirsch is the author of "Numerical Computation of Internal and External Flows Volume 1: Fundamentals of Numerical Discretization," 1988, and "Volume 2: Computational Methods for Inviscid and Viscous Flows," 1990, John Wiley and Sons Ltd, Chichester, UK. He has 20 book publications as editor and over 350 journal and conference publications. He has held several noteworthy memberships and distinctions including: Member of the Flemish Royal Academy of Science, Literature and Fine Arts (1995), and past Chairman of the European Research Community on Flow, Turbulence, and Combustion—ERCOFTAC (1988–1994). He is currently an ASME member.*