Available online at www.sciencedirect.com
ScienceDirect
Nuclear Engineering and Design 238 (2008) 716–743
Nuclear Engineering and Design
www.elsevier.com/locate/nucengdes

# Verification and validation benchmarks

William L. Oberkampf [a,*], Timothy G. Trucano [b]

[a] *Validation and Uncertainty Estimation Department, Sandia National Laboratories, Albuquerque, NM 87185-0828, USA*
[b] *Optimization and Uncertainty Estimation Department, Sandia National Laboratories, Albuquerque, NM 87185-0819, USA*

Received 5 December 2006; received in revised form 23 January 2007; accepted 26 February 2007

**Abstract**

Verification and validation (V&V) are the primary means to assess the accuracy and reliability of computational simulations. V&V methods and procedures have fundamentally improved the credibility of simulations in several high-consequence fields, such as nuclear reactor safety, underground nuclear waste storage, and nuclear weapon safety. Although the terminology is not uniform across engineering disciplines, code verification deals with assessing the reliability of the software coding, and solution verification deals with assessing the numerical accuracy of the solution to a computational model. Validation addresses the physics modeling accuracy of a computational simulation by comparing the computational results with experimental data. Code verification benchmarks and validation benchmarks have been constructed for a number of years in every field of computational simulation. However, no comprehensive guidelines have been proposed for the construction and use of V&V benchmarks. For example, the field of nuclear reactor safety has not focused on code verification benchmarks, but it has placed great emphasis on developing validation benchmarks. Many of these validation benchmarks are closely related to the operations of actual reactors at near-safety-critical conditions, as opposed to being more fundamental-physics benchmarks. This paper presents recommendations for the effective design and use of code verification benchmarks based on manufactured solutions, classical analytical solutions, and highly accurate numerical solutions. In addition, this paper presents recommendations for the design and use of validation benchmarks, highlighting the careful design of building-block experiments, the estimation of experimental measurement uncertainty for both inputs and outputs to the code, validation metrics, and the role of model calibration in validation. It is argued that the understanding of predictive capability of a computational model is built on the level of achievement in V&V activities, how closely related the V&V benchmarks are to the actual application of interest, and the quantification of uncertainties related to the application of interest.


## 1. Introduction

### 1.1. Background

The importance of computer simulations in the design and performance assessment of engineered systems has increased dramatically during the last three decades. The systems of interest include existing or proposed systems that operate, for example, at design conditions, at off-design conditions, and at failure-mode conditions that apply in accident scenarios. The role of computer simulations is especially critical if we are interested in the reliability, robustness, or safety of high-consequence systems that cannot ever be physically tested in a fully representative environment. Examples of such systems are the catastrophic failure of a full-scale containment building for a nuclear power plant, the long-term underground storage of nuclear waste, and a nuclear weapon involved in a transportation accident. In many situations, it is even difficult to specify what a "representative environment" actually means in

*Abbreviations:* AIAA, American Institute of Aeronautics and Astronautics; ASC, Advanced Simulation and Computing; ASME, American Society of Mechanical Engineers; CAD, computer-aided-design; CFD, computational fluid dynamics; CS&E, computational science and engineering; CSNI, Committee on the Safety of Nuclear Installations; DoD, Department of Defense; ERCOFTAC, European Research Community on Flow, Turbulence, and Combustion; GCI, Grid Convergence Index; GNC, guidance, navigation, and control; IEEE, Institute of Electrical and Electronics Engineers; ISO, International Standards Organization; ISP, International Standard Problem; LES, large eddy simulation; LOCA, loss-of-coolant accident; MMS, method of manufactured solutions; NAFEMS, National Agency for Finite Element Methods and Standards; NNSA, National Nuclear Security Administration; ODE, ordinary differential equation; PDE, partial differential equation; PDF, portable document format; RANS, Reynolds-Averaged Navier-Stokes; SQE, software quality engineering; SRQ, system response quantity; SSB, strong-sense benchmark; UQ, uncertainty quantification; V&V, verification and validation; V&V&UQ, verification and validation and uncertainty quantification

* Corresponding author.
*E-mail addresses:* wloberk@sandia.gov (W.L. Oberkampf), tgtruca@sandia.gov (T.G. Trucano).

0029-5493/$ – see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.nucengdes.2007.02.032

a complex system. Computer simulations of high-consequence systems are increasingly being used in furthering our understanding of the systems' responses, in developing public policy, in preparing safety procedures, and in determining legal liability. Thus, as computer simulations are given a more central role in the decision-making process, we believe the credibility of the computational results must be raised to a higher level than what has previously been considered acceptable. From a historical perspective, we are in the early days of changing from an engineering culture where hardware is built, tested, and then fixed if broken, to a culture that is more and more reliant on computational simulation. To have justified confidence in this evolving culture, we must make major improvements in the transparency and maturity of the computer codes used, the clarity of the physics included and excluded in the modeling, and the comprehensiveness of the uncertainty assessment performed. Stated more bluntly, we need to move from a culture of glossy marketing and arrogance to a culture that forthrightly addresses the limitations, weaknesses, and uncertainty of our simulations.

Developers of computational software, computational analysts, and users of the computational results face a critical question: How should confidence in computational science and engineering (CS&E) be critically assessed? Verification and validation (V&V) of computational simulations are the major processes for assessing and quantifying this confidence. Briefly, verification is the assessment of the software correctness and numerical accuracy of the solution to a given computational model. Validation is the assessment of the physical accuracy of a computational model based on comparisons between computational simulations and experimental data. In verification, the association or relationship of the simulation to the real world is *not* an issue. In validation, the relationship between computation and the real world (experimental data) *is* the issue.

The nuclear reactor safety community has a long history of contributing to the intellectual foundations of both V&V and uncertainty quantification (UQ). The risk assessment community in its studies and analysis of the underground storage of nuclear waste has also made significant contributions to the field of UQ. However, contributions from both of these communities to V&V&UQ have concentrated on software quality engineering (SQE) procedures, as well as on statistical procedures for risk assessment. 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 terminology, methodology, and procedures for V&V. The (only) book in the field on V&V provides a good summary of the development of many of the methodologies and procedures in computational fluid dynamics (CFD) (Roache, 1998a). In addition, Aeschliman and Oberkampf (1998), Oberkampf and Blottner (1998), Oberkampf and Trucano (2002), Oberkampf et al. (2004) provide a comprehensive review of the history and development of V&V from the perspective of the CFD community.

Achieving the next level of credibility in computational simulations will require concerted and determined efforts by individuals, universities, corporations, governmental agencies, commercial code-development companies, engineering societies, and standards-writing organizations throughout the world. The goal of these efforts should be to improve the reliability of the computer software, the estimation of numerical accuracy, the quality of the physics models used, the quantification of uncertainty, and the training and expertise of users of the codes. In addition, new methods are critically needed for effectively communicating the maturity and reliability of each of these elements, especially in relationship to decision making for high-consequence systems. This paper focuses on one aspect of the needed improvements to software reliability and physics modeling, namely, the construction and use of highly demanding V&V benchmarks. The benchmarks of interest are those related to the accuracy and reliability of physics models and codes. We are not interested here in benchmarks that relate to computer performance issues, such as the computing speed of codes on different types of computer hardware and operating systems.

During the last two decades, the National Agency for Finite Element Methods and Standards (NAFEMS) has developed some of the most widely known V&V benchmarks (NAFEMS, 2006). Roughly 30 verification benchmarks have been constructed by NAFEMS. The majority of these benchmarks have targeted solid mechanics simulations, though some of the more recent benchmarks have been in fluid dynamics. Most of the NAFEMS verification benchmarks consist of an analytical solution or an accurate numerical solution to a simplified physical process described by a partial differential equation (PDE). The NAFEMS benchmark set is carefully defined, numerically demanding, and well documented. However, these benchmarks are currently very restricted in their coverage of various mathematical and/or numerical difficulties and in their coverage of physical phenomena. Further, the performance of a given code on the benchmark is subject to interpretation by the user of the code. It is also likely that the performance of a code on the benchmark is dependent on the experience and skill of the user.

Several large commercial code companies specializing in solid mechanics have developed an extensive set of well-documented verification benchmarks that can be exercised by licensed users of their codes. Such benchmarks are intended to be applied only to a particular code, and they describe how that code performed on the benchmark problems. The performance results of a code tested on the benchmark problems by a commercial company can be clearly compared with the results obtained by a user who tests the code with the same benchmark problems. These company- and user-testing activities give the user a better understanding of the minimal performance that can be expected from a code. It should be noted here that information about a code's performance on a set of benchmark problems prior to purchase of the code is often difficult to obtain, as this information is proprietary.

Two examples of commercial codes with well-documented verification benchmarks are ANSYS® and ABAQUS®. ANSYS (2005) and ABAQUS (2006) have roughly 270 formal verification test cases. The careful description and documentation of the ANSYS and ABAQUS benchmark sets is impressive. However, the primary goal in essentially all of these documented benchmarks is to demonstrate the "engineering accuracy" of the codes, not to precisely and carefully quantify the numerical error

in the solutions. As stated in one set of documentation, "In some cases, an exact comparison with a finite-element solution would require an infinite number of elements and/or an infinite number of iterations separated by an infinitely small step size. Such a comparison is neither practical nor desirable" (ANSYS, 2005). We disagree completely with this point of view because (a) an exact comparison with a finite element solution does not require an infinite number of elements or iterations, or an infinitely small time step; and (b) it is practical and desirable to carefully assess the accuracy of a code by comparison with theoretically demanding solutions. Our support for these two counterarguments is expressed in the body of this paper.

Noticeably absent from the discussion of commercial codes above are CFD software packages. Although we have not surveyed all the major commercial CFD codes available, we have not found extensive, formally documented verification or validation benchmark sets for those codes we have examined. As an indication of the poor state of maturity of CFD software, a recent paper by Abanto et al. (2005) tested three unnamed commercial CFD codes on relatively simple verification test problems. The poor results of the codes were shocking to some people, but not to the authors of the paper and not to us.

In the field of nuclear reactor engineering, the Nuclear Energy Agency, Committee on the Safety of Nuclear Installations (CSNI) devoted significant resources toward developing validation benchmarks, which they refer to as International Standard Problems (ISPs). This effort began in 1977 with recommendations for the design, construction, and use of ISPs for loss-of-coolant accidents (LOCAs) (NEA, 1977). The CSNI recognized the importance of issues such as (a) providing a detailed description of the actual operational conditions in the experimental facility, not those conditions that were requested or desired; (b) preparing careful estimates of the uncertainty in experimental measurements and informing the analyst of the real estimate; (c) reporting the initial and boundary conditions that were realized in the experiment, not those conditions that were desired; and (d) conducting a sensitivity analysis to determine the most important factors that affect the predicted system responses of interest. The CSNI has continually refined the guidance for ISPs such that the most recent recommendations for the ISPs address any type of experimental benchmark, not just benchmarks for LOCA accidents (NEA, 2004). Thus, the primary goal of the ISPs remains the same for all types of benchmarks: "to contribute to a better understanding of postulated and actual events" that could affect the safety of nuclear power plants.

A number of efforts have been undertaken in the development of validation databases that could mature into well-founded benchmarks. In the United States, the NPARC Alliance has developed a validation database that has roughly 20 different flows (NPARC, 2000). In Europe, starting in the early 1990s, there has been a much more organized effort to develop validation databases. These databases have primarily focused on aerospace applications. The European Research Community on Flow, Turbulence and Combustion (ERCOFTAC) has collected a number of experimental datasets for validation applications (ERCOFTAC, 2000). QNET-CFD is a thematic network on quality and trust for the industrial applications of CFD (QNET-CFD, 2001). 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. For a history and review of the various efforts, see Rizzi and Vos (1998) and Vos et al. (2002).

We note that the validation databases described by Rizzi and Vos (1998) and Vos et al. (2002) contain many cases that are for very complex flows, which are sometimes referred to as "industrial applications." We have observed, however, both through our own experience and in the open literature, that attempts to validate models on complex physical processes are commonly unsuccessful because the computational results do not compare well with the experimental measurements. Then the computational analysts often do one of the following: (1) they engage in a model calibration activity, dealing with both physical and numerical parameters in the model, to obtain better agreement; (2) they reformulate the assumptions in their model to obtain better agreement, thereby changing the model; or (3) they start pointing accusatory fingers at the experimentalists about either what is wrong with the experimental data or what the experimentalists should have measured to make the data more effective for validation. Regarding model calibration specifically, we view this activity as a useful and pragmatic path forward for application of the calibrated model in future predictions that are very similar to the experimental database. Calibration, however, rarely addresses the underlying weaknesses of the models because typically there are so many modeling approximations, or deficiencies, that could be contributing to the disagreement (Oberkampf and Trucano, 2002). We believe that calibration should be undertaken when it is clearly understood that this activity is a response to V&V assessment, not a replacement for V&V assessment (AIAA, 1998; ASME, 2006; Trucano et al., 2006).

As we discuss in more detail in Section 2.2, validation benchmarks are much more difficult to construct and use than verification benchmarks. The primary difficulty in constructing validation benchmarks is that experimental measurements in the past have rarely been designed to provide true validation benchmark data. Oberkampf and Aeschliman (1992), Aeschliman and Oberkampf (1998), Oberkampf and Blottner (1998), Oberkampf and Trucano (2002), and Roy et al. (2003a,b) give an in-depth discussion of the characteristics of validation experiments, as well as an example of a wind tunnel experiment that was specifically designed to be a true validation benchmark. The validation benchmarks that have been compiled and documented by organized efforts, some of which were referenced above, are indeed instructive and useful to users of the codes and to developers of physics models. However, we argue in this paper that much more needs to be incorporated into the validation benchmarks, both experimentally and computationally, to achieve the next level of usefulness and critical assessment.

Oberkampf et al. (2004) introduced the concept of *strong-sense benchmarks* (SSBs) in V&V. Oberkampf et al. argued that SSBs should be of a high-enough quality that they can be viewed as *engineering reference standards*. These authors stated that SSBs are test problems that have the following four character-

istics: (1) the purpose of the benchmark is clearly understood, (2) the definition and description of the benchmark is precisely stated, (3) specific requirements are stated for how comparisons are to be made with the results of the benchmark, and (4) acceptance criteria for comparison with the benchmark are defined. In addition, these authors required that information on each of these characteristics be "promulgated," i.e., the information is well documented and publicly available. Although a number of benchmarks are available, a few of which were discussed previously, these authors asserted that SSBs do not presently exist in computational physics or engineering. They suggested that professional societies, academic institutions, governmental or international organizations, and newly formed nonprofit organizations would be the most likely to construct SSBs. This paper builds on these basic ideas and provides detailed recommendations for the characteristics of V&V SSBs and suggestions on how computational simulations can be compared with SSBs.

### 1.2. Outline of the paper

Section 2 begins with a brief review of the terminology of both verification and validation and points out how different communities have varying interpretations of these processes. The two types of verification, code verification and solution verification, are then discussed. It is pointed out that validation is composed of three quite different activities: assessment of the accuracy of computational models by comparison with experiments; extrapolation of these models to applications of interest; and determination if the estimated accuracy of the extrapolation is adequate for the applications of interest. The concept of a validation hierarchy is discussed, which is a valuable tool for assessing the accuracy of computational models at many different levels of complexity. The section ends with a focus on validation experiments, identifying the required characteristics of these experiments and explaining how these experiments differ from traditional experiments and how they form the central role in the construction of validation benchmarks.

Section 3 discusses our recommendations for constructing and using verification benchmarks. First, we present the four elements that should be contained in the documentation of a verification benchmark: (1) conceptual description, (2) mathematical description, (3) accuracy assessment, and (4) additional user information. Examples are provided for applying these elements to the four types of benchmarks, namely, manufactured solutions, analytical solutions, numerical solutions to ordinary differential equations (ODEs), and numerical solutions to PDEs. We recommend that when a candidate code is compared with a verification benchmark, the results of the comparisons with benchmarks not be included in the benchmark documentation per se. We next discuss how formal comparison results could be used and identify the types of information that should be included in the comparisons.

Section 4 discusses our recommendations for constructing and using validation benchmarks. First, we present the four elements that should be contained in the documentation of a validation benchmark: (1) conceptual description; (2) experimental description; (3) uncertainty quantification of benchmark measurements; and (4) additional user information. We next discuss how candidate code results could be compared with the benchmark results, paying particular attention to issues related to the computation of nondeterministic results to determine the uncertainty of system response quantities (SRQs) due to uncertainties in input quantities, the computation of validation metrics to quantitatively measure the difference between experimental and computational results, the minimization of model calibration in comparisons with validation benchmarks, and the constructive role of global sensitivity analyses in validation experiments.

Section 5 raises a diverse set of issues about how a V&V benchmark database might be initiated and implemented, as well as be a contributor to CS&E. Examples of these issues include the following: primary and secondary goals of the database, initial construction of an Internet-based system, software construction of the database, review and approval procedures for entries into the database, open versus restricted use of the database, organizational control of the database, and funding of the database.

Closing remarks and some possible implications of constructing a V&V benchmark database are given in Section 6.

## 2. Review of verification and validation processes

Various technical disciplines have long had varying definitions for verification and validation. The Institute of Electrical and Electronics Engineers (IEEE) was the first major engineering society to develop formal definitions for V&V (IEEE, 1984). These definitions, initially published in 1984, were adopted by the American Nuclear Society (ANS, 1987) and the International Standards Organization (ISO) (ISO, 1991). After a number of years of discussion and intense debate in the U.S. defense and CFD communities, the IEEE definitions were found to be confusing and lacking in utility. In particular, these definitions did not directly address certain issues that are very important in CS&E, such as the dominance of algorithmic issues in the numerical solution of PDEs, and the importance of comparisons of computational results with the "real world." As a result, the U.S. Department of Defense (DoD) developed an alternate set of definitions (DoD, 1996a,b). Following very closely the DoD definitions, the American Institute of Aeronautics and Astronautics (AIAA) and the American Society of Mechanical Engineers (ASME) adopted the following definitions (AIAA, 1998; ASME, 2006):

- V*erification*: The process of determining that a model implementation accurately represents the developer's conceptual description of the model and the solution to the model.
- V*alidation*: 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.

These definitions have also been recently adopted by the U.S. Department of Energy National Nuclear Security Administration's (NNSA's) Advanced Simulation and Computing (ASC) program (Soudah et al., in preparation). For a detailed discussion of the history of the development of the terminology from the

perspective of the CS&E communities, see Oberkampf (1994), Mehta (1995), Oberkampf and Trucano (2002), and Oberkampf et al. (2004).

Verification provides evidence, or substantiation, that the mathematical model, which is derived from the conceptual model, is solved correctly by the computer code that is being assessed. In CS&E, the mathematical model is typically defined by a set of partial differential or integro-differential equations, along with the required initial and boundary conditions. The computer code solves the computational model, i.e., the discrete-mathematics version (or mapping) of the mathematical model translated into software. The fundamental strategy in verification is to identify, quantify, and reduce errors caused by the mapping of the mathematical model to a computer code. Verification does not address the issue of whether the mathematical model has any relationship to the real world, e.g., physics.

Validation, on the other hand, provides evidence, or substantiation, of how accurately the computational model simulates the real world for system responses of interest. The U.S. DoD and many other organizations must deal with complex systems composed of physical processes, computer-controlled subsystems, and strong human interactions. From the perspective of these organizations, assessment of accuracy compared to the real world would include expert opinion and well-founded results from other computer simulations. From the perspective of the CS&E community, the real world is traditionally viewed to *only* mean experimentally measured quantities in a physical experiment (AIAA, 1998; ASME, 2006). Validation activities presume that the computational model result is an accurate solution of the mathematical model. However, programming errors in the computer code, deficiencies in the numerical algorithms, or inaccuracies in the numerical solution, for example, may cancel one another in specific validation calculations and give the illusion of an accurate representation of the experimental measurements. Verification, thus, should ideally be accomplished *before* the validation comparisons are made so that one's assessment of numerical accuracy is not influenced by whether the agreement of the computational results with experimental data is "good" or "bad." While verification is not simple, it is conceptually less complex than validation because it deals with mathematics and computer science issues. Validation, on the other hand, must address a much broader range of issues: assessment of the fidelity of the mathematical modeling of physical processes; assessment of the consistency, or relevance, of the mathematical model to the physical experiment being conducted; influence of the experimental diagnostic techniques on the measurements themselves; and estimation of experimental measurement uncertainty. Validation rests on evidence that the appropriate experiments were executed correctly, as well as on evidence that supports the mathematical accuracy of the computed solution. These issues are practically coupled in nontrivial ways in complex validation problems although they are logically distinct. As Roache (1998a) succinctly states, "Verification deals with mathematics; validation deals with physics."

### 2.1. Verification activities

#### 2.1.1. Fundamentals of verification

Two types of verification are generally recognized and defined in computational simulation: code verification and solution verification (Roache, 1998a,b). Recent work by Oberkampf and Trucano (2002) argues that it is useful to further segregate code verification into two activities: numerical algorithm verification and software quality engineering (SQE), as shown in Fig. 1. Numerical algorithm verification addresses the mathematical correctness of the software implementation of all the numerical algorithms that affect the numerical accuracy of the computational results. 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. The emphasis in SQE is on determining whether or not the code, as part of a software sys-

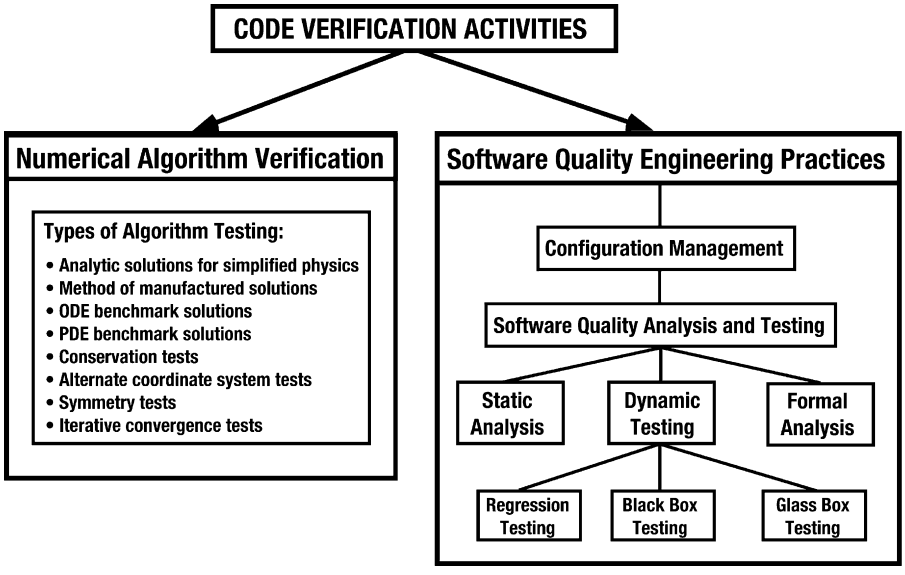


Fig. 1. Integrated view of code verification in computational simulation (Oberkampf et al., 2004).

tem, is reliable (implemented correctly) and produces repeatable results on specified computer hardware and in a specified software environment, including compilers, libraries, and so forth. SQE procedures are primarily needed during software development, testing, and modification.

Numerical algorithm verification, SQE, and solution verification, are fundamentally empirical. Specifically, these activities are based on observations, comparisons, and analyses of the code results for individual executions of the code. Numerical algorithm verification focuses on 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 because it is the code itself that must *demonstrate* the analytical and formal results of the numerical analysis. Numerical algorithm verification is usually conducted by comparing computational solutions with highly accurate solutions.

Fig. 1 depicts a top-down process with two main branches of code verification: numerical algorithm verification and SQE practices (Oberkampf et al., 2004). Numerical algorithm verification, discussed in Section 2.1.2, focuses on accumulating 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 algorithm testing or simply as code verification. SQE activities include practices, procedures, and processes that are primarily developed by researchers and practitioners in the computer science and IEEE communities. Conventional SQE emphasizes processes (management, planning, acquisition, supply, development, operation, and maintenance), as well as reporting, administrative, and documentation requirements. A key element, or process, of SQE is software configuration management, which is composed of configuration identification, configuration and change control, and configuration status accounting. These three activities are primarily directed toward programming correctness in the source program, system software, and compiler software. As shown in Fig. 1, SQE and testing can be divided into static analysis, dynamic testing, and formal analysis. Dynamic testing can be further divided into such elements of common practice as regression testing, black-box testing, and glass-box testing. From an SQE perspective, Fig. 1 could be reorganized such that all the types of algorithm testing listed on the left, under numerical algorithm verification, could be moved under dynamic testing. However, the computer science and IEEE communities have shown no formal interest in the development of the testing procedures listed under numerical algorithm verification. These testing procedures, on the other hand, dominate code development practice in the traditional CS&E communities.

Unfortunately, as discussed in Trucano et al. (2005), when solving complex PDEs, a computational scientist finds it virtually impossible to decouple the distinct problems of mathematical correctness, algorithm correctness, and software-implementation correctness. For instance, algorithms often represent nonrigorous mappings of the mathematical model to the underlying discrete equations. Two examples of such mappings are (1) approximate factorization of difference operators; and (2) algorithms that are derived assuming high levels of smoothness of the dependent variables in the PDEs, when in reality the algorithms are applied to problems with little or no continuity of the derivatives of the variables. Whether such algorithms produce correct solutions to the PDEs cannot be assessed without executing the code on specific problems; the execution of the code is, in turn, coupled to the software implementation. One consequence of these couplings among mathematics, algorithms, and the software implementation is that the source of a numerical inaccuracy cannot be easily identified. These couplings also suggest that there is a greater overlap between PDE complexities, discrete mathematics, and SQE than some practitioners might prefer.

Solution verification centers on 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 both numerical algorithm verification and SQE, we believe solution verification could also be referred to as *numerical error estimation*. That is, the primary goal of solution verification is to estimate the numerical accuracy of a given solution, typically for a nonlinear PDE with singularities and discontinuities. The assessment of numerical accuracy is a key activity in computations used for validation, as well as those generated for specific applications. Numerical error estimation is strongly dependent on the quality and completeness of code verification.

The two basic approaches for estimating the error in a numerical solution to a PDE are *a priori* and *a posteriori* error estimation techniques. An *a priori* approach only uses 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 finite element methods and finite volume methods (Hirsch, 1988, 1990; Oden, 1993; Ferziger and Peric, 1996; Morton, 1996; Laney, 1998; Roache, 1998a,b). An *a posteriori* approach can use all the *a priori* information as well as the computational results from previous numerical solutions, e.g., solutions using different mesh resolutions or solutions using different order-of-accuracy methods. We believe that the only way to achieve a quantitative estimate of numerical error in practical cases of nonlinear, complex PDEs is by using *a posteriori* error estimates.

*A posteriori* error estimation has primarily been performed through the use of either Richardson extrapolation (Roache, 1998a,b) or more sophisticated estimation techniques that are based on finite element approximations (Ainsworth and Oden, 2000; Babuska and Strouboulis, 2001). Richardson extrapolation uses solutions on a sequence of carefully constructed meshes having different levels of mesh refinement to estimate the spatial discretization error. This method can also be used on a sequence of solutions with varying time-step increments to estimate the temporal discretization error. Richardson's method can be applied to any discretization procedure for differential or integral equations, e.g., finite difference methods, finite element methods, finite volume methods, spectral methods, and

boundary element methods. As Roache (1998a,b) points out, Richardson's method produces different estimates of error and uses different norms than the traditional *a posteriori* error methods used in finite elements (Hirsch, 1990; Roache, 1990). The Grid Convergence Index (GCI) method, based on Richardson's extrapolation, was developed by Roache to assist in the estimation of mesh resolution error (Roache, 1994, 1997, 1998a,b).

### 2.1.2. Code verification procedures

Considering the numerical solution of PDEs, code verification comprises the activities of (1) defining appropriate benchmarks for the evaluation of solution accuracy and (2) determining what constitutes satisfactory performance of the algorithms on the benchmarks. Code verification relies on the comparison of computational solutions to the "correct answer." The correct answer is provided by highly accurate solutions for a set of well-chosen benchmarks, and this answer can only be known in a relatively small number of isolated cases. These cases therefore assume a very important role in code verification and should be carefully formalized in test plans that describe how the code will be verified.

Fig. 2 depicts a method that uses exact or highly accurate solutions to the PDEs to detect numerical algorithm deficiencies and programming errors. The conceptual model is constructed by (1) considering the important physics of interest that are relevant to the system being analyzed and (2) determining the system response quantities (SRQs) that are needed for the application of interest. The mathematical model is derived from the conceptual model. The mathematical model is typically given by a set of PDEs and all their associated input data, e.g., initial conditions, boundary conditions, and material properties. The mathematical model is the general model for the application of interest, whereas the exact and highly accurate solutions to the PDEs are special-case solutions of the mathematical model. For these special cases, benchmark solutions can be computed.

The equations in the mathematical model are discretized, i.e., mapped from derivatives and integrals to algebraic equations, and solution procedures are developed using the selected numerical algorithms. The discretized equations are then programmed in the computer code, creating a computational model. When the computational model is executed to solve the benchmark problem, the model produces the computational results of interest. The computational results are then compared with the benchmark solution results, and any differences between the two results are evaluated. Comparisons are typically made for various SRQs of interest. The comparisons are usually examined along boundaries of the solution domain common or error norms are computed over the entire solution domain so that the accuracy of various SRQs can be determined.

Probably the most important challenge in the design and computation of verification benchmarks for use in the process depicted in Fig. 2 is to assess the mathematical accuracy of the benchmark solution. The AIAA Guide (AIAA, 1998) suggests the following hierarchical organization with respect to the accuracy of benchmark solutions (from highest to lowest): analytical solutions, highly accurate numerical solutions to the ODEs, and highly accurate numerical solutions to the PDEs. In the AIAA Guide, as well as in Oberkampf et al. (2004), analytical solutions included manufactured solutions that were constructed by the "Method of Manufactured Solutions" (MMS) (Roache, 1998a,b). Recently, however, the present authors have concluded that the manufactured solutions should be considered as a separate type of highly accurate solutions. This conclusion was based on two reasons: (a) manufactured solutions do not correspond to physically meaningful phenomena, and (b) they do not suffer from numerical accuracy issues that commonly occur with analytical solutions. Thus, the hierarchical organization presented in this paper is expanded to include the following four types of highly accurate solutions (from highest to lowest): (type 1)

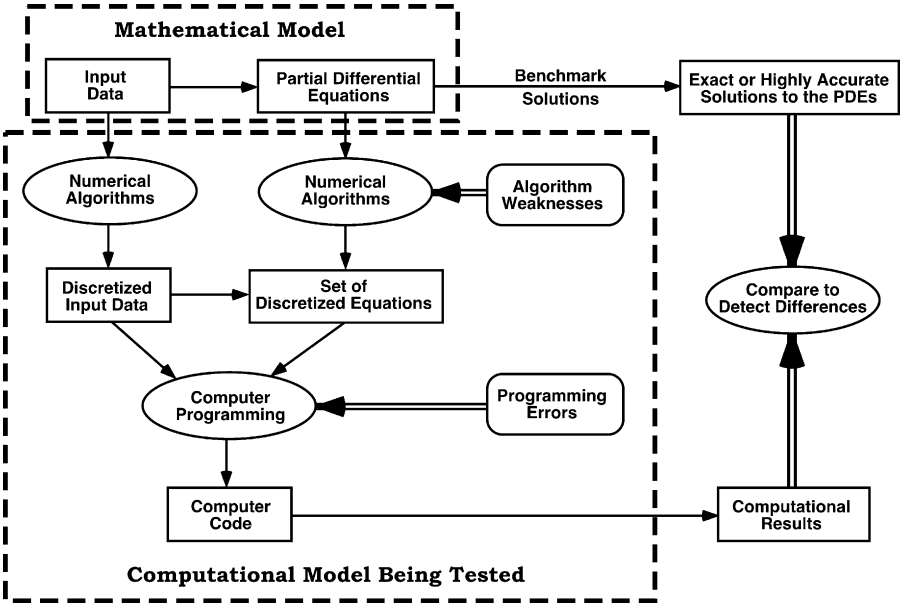


Fig. 2. Method to detect sources of errors in code verification.

manufactured solutions, (type 2) analytical solutions, (type 3) highly accurate numerical solutions to the ODEs, and (type 4) highly accurate numerical solutions to the PDEs. These types are discussed briefly below, though additional detail is given for manufactured solutions because they are not widely used and not widely understood.

Manufactured solutions (type 1) are specifically constructed for testing numerical algorithms and computer codes (Roache, 1998a,b; Knupp and Salari, 2002). The MMS allows one to custom-design verification solutions by altering the original PDEs of interest in the mathematical model (Fig. 2). A specific form of the solution function is chosen and then the original PDE of interest is modified such that the chosen solution function satisfies the modified PDE. The solution function is inserted into the original PDE, and all the derivatives are obtained through symbolic manipulation. Typically, these derivatives are obtained by using symbolic manipulation software such as MATLAB® 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, on the right-hand side of the PDE. With this new source term, the assumed solution function satisfies the new PDE exactly. When this source term is added to the original PDE, one recognizes that we are no longer dealing with physically meaningful phenomena, although we remain in the domain of mathematical interest. This realization can cause some researchers or analysts to claim that the solution is no longer relevant to computational simulation. The fallacy of this argument is apparent by noting that in verification we are only dealing with testing the numerical algorithms and with coding, *not* the relationship of the code results to physical responses of the system. Because the solution to the modified PDE was "manufactured," the boundary conditions for the new PDE are analytically derived from the chosen solution. For the three types of common boundary conditions, one can use the chosen solution function to (a) simply evaluate the solution on any boundary of interest, i.e., a Dirichlet condition; (b) analytically derive a Neumann type boundary condition and apply it on any boundary; and (c) analytically derive a boundary condition of the third kind and apply it on any boundary. The MMS is appropriately described as finding the problem, i.e., the PDE, for which a solution has been assumed.

Using the MMS in code verification requires that the analytically derived source term, containing only algebraic expressions, be inserted into the code being tested. The MMS verifies many numerical aspects in the code, such as the mathematical correctness of the numerical algorithms, the spatial-transformation technique for grid generation, the grid-spacing technique, and the absence of coding errors in the software implementation. As pointed out by a number of researchers of this topic, e.g., Roache (1998a,b), and Knupp and Salari (2002), solutions in the MMS must be carefully chosen to achieve the desired test results. For example, solution forms should be chosen so that as many terms as possible in the original PDE produce nonzero values during the computation of the solution. Such terms could include submodels that are part of the set of PDEs, as well as any mathematical transformations of physical space to computational space.

Analytical solutions (type 2) are closed-form solutions to special cases of the PDEs defined in the mathematical model. These closed-form solutions are commonly represented by infinite series, complex integrals, and asymptotic expansions. Numerical methods having known reliability and accuracy must be used to compute the infinite series, complex integrals, and asymptotic expansions to obtain the solutions of interest. The accuracy of these solutions, particularly if they are infinite series or asymptotic expansions, must be carefully quantified; and quantifying the accuracy of the solutions can be very challenging. The most significant practical shortcoming of classical analytical solutions is that they exist only for very simplified physics, material properties, and geometries.

The third type of highly accurate solutions consists of numerical solutions to special cases of the general PDEs that can be mathematically simplified to ODEs. The ODEs can be either initial value problems or two-point boundary value problems. The ODEs commonly result from simplifying assumptions to the original PDEs. For example, we may make the assumptions that are needed to simplify the original PDEs given in three dimensions so that one obtains one-dimensional ODEs. Another example is to use simple geometries that allow similarity variables to be constructed for the original PDE, resulting in an ODE. Once an ODE has been obtained, a highly reliable and accurate ODE solver must then be used to compute the numerical solution.

In fluid dynamics, some well-known ODE benchmarks are stagnation point flow, specialized cases of laminar flow in two dimensions, the Taylor-Maccoll solution for inviscid flow over a sharp cone, and the Blasius solution for laminar flow over a flat plate. Note that the Blasius solution would be a useful benchmark for assessing the accuracy of a CFD code that solves the boundary layer equations. However, the Blasius solution would *not* be a good benchmark for testing a Navier-Stokes code because the Blasius solution also relies on the approximations assumed in the boundary layer theory. There is a difference between a highly accurate Blasius solution and a highly accurate Navier-Stokes solution because of the different assumptions made in the two physics models. The modeling assumptions *must* be the same between the benchmark solution and the code being tested. Some argue that the solutions obtained from two closely related physics models may be "adequate." However, when small differences in solutions exist, one cannot distinguish between slight differences due to modeling assumptions versus a coding error.

The fourth type of highly accurate solutions consists of numerical solutions to more complex PDEs, i.e., more complex than the three types just discussed. The accuracy of numerical solutions to more complex PDEs clearly becomes more questionable when such solutions are compared with manufactured solutions, analytical solutions, or ODE solutions. The numerical reliability of a type four solution is itself a factor that is hard to separate from the verification task the benchmark is intended to perform. In the literature, for example, one can find descriptions of computational simulations that are considered to be "benchmark solutions" by the author, but that are later found by other researchers to be lacking. And although it is common practice to conduct code-to-code comparisons, we argue that these kinds

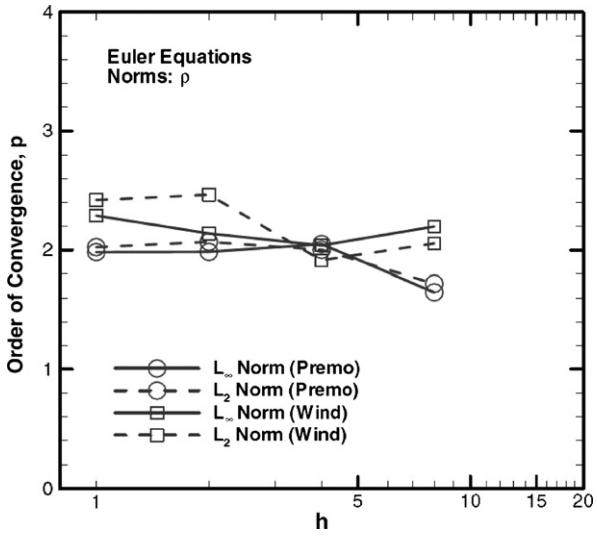


Fig. 3. Observed order of convergence as a function of mesh resolution for two Navier-Stokes codes (Roy, 2004).

of comparisons are of limited value *unless* highly demanding requirements are imposed on the numerical solution that is considered to be the "benchmark" (Trucano et al., 2003). These requirements are discussed in detail in Section 3.1.

In code verification, the key feature to determine is the observed, or demonstrated, order of convergence using multiple numerical solutions. As discussed in Roache (1998a,b), and Knupp and Salari (2002), Richardson extrapolation is used in combination with the known exact solution and results from two different mesh resolutions to determine the observed order of convergence from a code. A typical plot of observed order of convergence versus mesh resolution is shown in Fig. 3. When the mesh is well-resolved in the spatial dimension, the numerical solution enters the asymptotic convergence region. In this region, the observed order of convergence becomes approximately constant, meaning that the error decreases at a fixed rate as the mesh is further resolved. By computing the observed order of convergence in testing a code, an analyst can make two strong statements about accuracy. First, if the observed order is greater than zero, then the code converges to the correct solution as the mesh is refined. If the observed order of convergence is zero, then the code will *not* converge to the correct answer. Second, if the observed order of convergence matches (or nearly matches) the formal order of convergence, then the code demonstrates that it can reproduce the theoretical order of convergence of the numerical method. The theoretical order of convergence of a complex set of numerical algorithms may actually not be known rigorously, or it may be the case that the scheme is a mixed-order scheme. For complex algorithms, special techniques must then be employed when using the MMS (Roache, 1998a,b; Knupp and Salari, 2002).

Researchers have found a number of reasons why the observed order of convergence can be less than the formal accuracy when the latter is rigorously known. Some of the reasons are as follows: (1) a programming error exists in the computer code; (2) the numerical algorithm is deficient is some unanticipated way; (3) there is insufficient grid resolution such that the grid is not in the asymptotic convergence region of the power-series expansion for the particular SRQ of interest; (4) the formal order of convergence for interior grid points is different from the formal order of convergence for boundary conditions involving derivatives, resulting in a mixed order of convergence over the solution domain; (5) singularities, discontinuities, and contact surfaces are interior to the domain of the PDE; (6) singularities and discontinuities occur along the boundary of the domain; (7) the mesh resolution changes abruptly over the solution domain; (8) there is inadequate convergence of an iterative procedure in the numerical algorithm; and (9) boundary conditions are over-specified. It is beyond the scope of this paper to discuss the reasons listed above in detail; however, some of the representative references in these topics are Keller (1969), Srivastava et al. (1979), Blottner (1982), Turkel (1986), Axelsson (1996), Ferziger and Peric (1996), Roache (1998a,b), Carpenter and Casper (1999), Roy et al. (2000), Botella and Peyret (2001), Roy and Blottner (2001), Diskin and Thomas (2002), and Knupp and Salari (2002).

### 2.2. Validation activities

#### 2.2.1. Fundamentals of validation

Some researchers and engineering standards documents (AIAA, 1998; Pace, 1998; Oberkampf and Trucano, 2002; Oberkampf et al., 2004; ASME, 2006; Trucano et al., 2006) have identified three key, and distinct, issues in validation: (1) quantification of the accuracy of the computational model by comparing its responses with experimentally measured responses, (2) interpolation or extrapolation of the computational model to conditions corresponding to the intended use of the model, and (3) determination if the estimated accuracy of the computational model, for the conditions of the intended use, satisfies the accuracy requirements specified. The definition of validation, given at the beginning of Section 2, is not particularly clear, however, about the identification of these issues. Consequently, this definition of validation can be interpreted to include all three issues, or interpreted to only include the first issue. Fig. 4 depicts these three issues, as well as the input information required by these issues.

It is clear from Fig. 4 that the quantification of model accuracy (issue 1) obtained by comparing responses from the computational model with experimentally measured responses is distinctively different from prediction, e.g., extrapolation of the model beyond the domain of validation to the conditions of the intended use (issue 2). The interpolation or extrapolation of the model for its intended use must include the estimated uncertainty in the prediction, which is then compared with the accuracy requirements so that a decision can be made whether the prediction accuracy is adequate (issue 3). The most recent engineering standards document devoted to V&V, referred to as the ASME Guide (ASME, 2006), considers all three aspects of validation to be fundamentally combined in the term "validation." The AIAA Guide (AIAA, 1998), on the other hand, takes the view that "validation" is only concerned with the first issue, i.e., assessment of model accuracy by comparison with experimental responses. Uncertainty is involved in this assess-

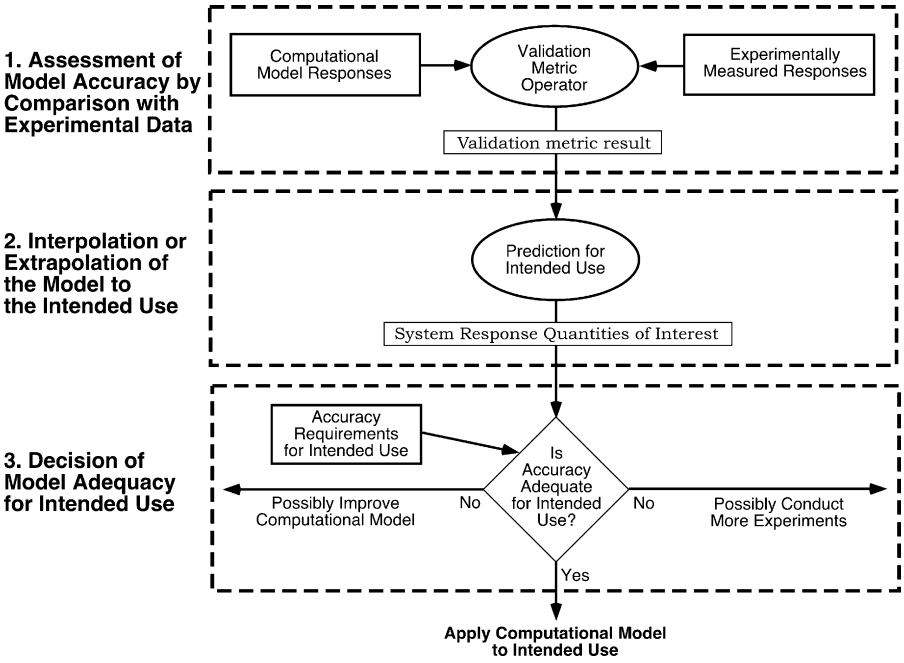


Fig. 4. Three aspects of model validation.

ment, both in terms of experimental measurement uncertainty and in terms of the computational simulation, primarily because input quantities needed from the experiment either are not available or are imprecisely characterized. The second and third aspects (issues 2 and 3) are treated in the AIAA Guide as separate activities related to predictive capability. The AIAA Guide recognizes that predictive capability uses the assessed model accuracy as input *and* that predictive capability also incorporates (a) additional uncertainty estimation resulting from interpolation or extrapolation of the model beyond the existing experimental database to future applications of interest; and (b) comparison of the accuracy requirements needed by a particular application relative to the estimated accuracy of the model for that specific extrapolation to the applications of interest.

The two perspectives of validation discussed above are useful and workable, but the formal terminology for validation clearly can mean different things. Thus, one must be very clear when speaking and writing on the subject of validation. As a separate topic, whether the system of interest, e.g., component of a nuclear power plant, meets its performance or safety requirements is, of course, a *completely* separate topic from the issues depicted in Fig. 4. Simply put, a model of a system could be accurate, but the system itself could fail to meet requirements.

The hydrology community (Rykiel, 1996; Beven, 2002; Refsgaard and Henriksen, 2004) in Europe have independently developed ideas about V&V that are very similar to those being developed in the United States. Rykiel (1996) makes an important practical point, especially to analysts and decision makers, about the difference between the philosophy-of-science viewpoint and the practitioner's view of validation; "Validation is not a procedure for testing scientific theory or for certifying the 'truth' of current scientific understanding .... Validation means that a model is acceptable for its intended use because it meets specified performance requirements." Refsgaard and Henriksen (2004) have recommended terminology and fundamental procedures for V&V that are applicable to a much wider range of simulations than just hydrological modeling. Their definition of validation makes the two aspects of validation in Fig. 4 clear; "Model Validation: Substantiation that a model within its domain of applicability possesses a satisfactory range of accuracy consistent with the intended application of the model." Refsgaard and Henriksen also have stressed another crucial issue that is corroborated by the AIAA Guide and the ASME Guide: "Validation tests against independent data that have not also been used for calibration are necessary in order to be able to document the predictive capability of a model." In other words, the major challenge in validation is to perform an assessment of the model in a "blind" test with experimental data, whereas the key issue in calibration is to adjust the physical modeling parameters to improve agreement with experimental data. It is difficult, and sometimes impossible, to make blind comparisons, e.g., when well-known benchmark validation data are available for comparison. However, we must be very cautious in making conclusions about the predictive accuracy of models when the analyst has seen the data. Knowing the "correct answer" beforehand is extremely seductive, even to a saint.

An additional fundamental, as well as practical, aspect of validation in a real engineering environment has been the construct of a validation hierarchy (AIAA, 1998; ASME, 2006). Because it is neither feasible nor practical to conduct true validation experiments on most complex or large-scale systems, the recommended method is to use a building-block approach. This approach divides the complex engineering system of interest into three or more progressively simpler levels of complexity (tiers): subsystem cases, benchmark cases, and unit problems. In

the reactor safety field, a similar concept, referred to as *separate effects testing*, has been used for a long time. The strategy in the tiered approach is to assess how accurately the computational responses compare with the experimental responses at multiple levels of physics coupling and geometric complexity. The tiered approach is very useful for several reasons: (1) the hierarchy can represent a large range of complexity in systems, physics, material, and geometry; (2) the hierarchy requires a wide range of experienced individuals to construct it, providing the opportunity for discovering subsystem or component interactions that had not been previously recognized; (3) the hierarchy supports testing of models, or submodels, at any of the tiers of complexity; and (4) different hierarchies would be constructed for analyzing the system under different environments, e.g., normal, abnormal, and hostile environments. In addition, the tiered approach recognizes that the quantity, accuracy, and cost of information that is obtained from experiments vary radically over the range of tiers.

Importantly, each comparison of computational responses and experimental responses in a validation hierarchy allows an inference of model accuracy to be made relative to the tiers that are immediately above and below the tier where the comparison is made. The construction and use of a validation hierarchy is particularly important in situations where the complete system of interest cannot be tested. For example, the nuclear power industry has used constructs like a validation hierarchy in safety studies and probabilistic risk assessment for abnormal environment scenarios.

#### 2.2.2. *Characteristics of validation experiments*

With the critical role that validation experiments play in the assessment of model accuracy and predictive capability, it is reasonable to ask what a validation experiment is and how a validation experiment is different from other experiments. In responding to such questions, we first 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, or discover new phenomena. Sometimes these are referred to as scientific discovery experiments. The second category consists of experiments that are conducted primarily for constructing or improving mathematical models of fairly well-understood physical processes. Sometimes these are referred to as model calibration experiments. The third category includes experiments that determine or improve the reliability, performance, or safety of components, subsystems, or complete systems. These experiments are sometimes called "proof tests" or "system performance tests."

The present authors and their colleagues (Oberkampf and Aeschliman, 1992; Walker and Oberkampf, 1992; Oberkampf et al., 1993, 1995; Aeschliman and Oberkampf, 1998; Oberkampf and Blottner, 1998; Trucano et al., 2002) have argued that validation experiments constitute a new type of experiment. A validation experiment is conducted for the primary purpose of determining the predictive accuracy of a computational model or group of models. In other words, a validation experiment is designed, executed, and analyzed for the purpose of quantitatively determining the ability of a mathematical model and its embodiment in a computer code to simulate a well-characterized physical process or set of processes. Thus, in a validation experiment "the code is the customer"; or, if you like, "the computational scientist is the customer." Only during the last 10–20 years has computational simulation matured to the point where it could even be considered as a customer in this sense. As modern technology increasingly moves toward engineering systems that are designed, and possibly even fielded, based predominately on CS&E, CS&E itself will increasingly become the customer of experiments.

During the past several years, a group of researchers at Sandia National Laboratories has been developing methodological guidelines and procedures for designing and conducting a validation experiment (Oberkampf and Aeschliman, 1992; Walker and Oberkampf, 1992; Oberkampf et al., 1993, 1995; Aeschliman and Oberkampf, 1998; Oberkampf and Trucano, 2002; Trucano et al., 2002; Roy et al., 2003a,b). These guidelines and procedures have emerged as part of a concerted effort in the NNSA ASC program to provide a rigorous foundation for V&V for computer codes that are important elements of the U.S. nuclear weapons program (Kusnezov, 2004). Historically, the guidelines presented below were first developed in their current form in a joint computational and experimental program conducted in a wind tunnel, though they apply to a wide range of CS&E.

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 guidelines above are applicable to any tier in the validation hierarchy discussed earlier. A detailed discussion of the six guidelines is beyond the scope of the present work. The reader is referred to the given references in the previous

paragraph for an in-depth discussion of what the guidelines mean, how they can be implemented, and the difficulties that may be encountered when conducting validation experiments. Some of these guidelines have been incorporated into the recommendations for the construction of validation benchmarks in Section 4.1.

More recent efforts have been made to optimize the effectiveness and value of validation experiments (Trucano et al., 2002). Our recommended approach consists of the following three strategies: (1) early in the planning process, define the goals and the expected results of the validation activity; (2) design the validation experiment by using the code in a predictive sense and also account for the limitations in capability of the experimental facility; and (3) develop a well-thought-out plan for analyzing and quantitatively comparing the computational and experimental responses.

The first strategy, defining the goals and expected results, deals with issues such as (a) determining how the validation activity relates to the application of interest (typically through the validation hierarchy); (b) identifying the physics modeling issues that will be tested; (c) deciding whether the validation activity is intended to severely test the model to identify its weaknesses or whether it is intended to make the model look good, for example, to a potential customer; (d) specifying what will be required from both the computational and experimental aspects of the validation activity to conclude that each aspect was deemed a "success"; and (e) defining the steps that would be taken if the computational results agree very poorly with the experimental measurements.

In the second strategy above, "design" means using the code to directly guide the design features of the experiment, including such elements as geometry, initial and boundary conditions, material properties, sensor locations, and diagnostic techniques (e.g., strain gauges, thermocouples, optical techniques, and radiation detectors). Even if the accuracy of the code predictions is not expected to be high, the code can frequently guide much of the design of the experiment. For example, such code involvement minimizes the risk that a validation experiment will produce measurements that cannot be synthesized by the computational model. The code and the goals of the validation activity can also guide the accuracy that is needed for the experimental measurements as well as the number of experimental realizations that are needed to obtain a specific statistically significant result. Suppose, through a series of exploratory calculations for a particular application of the code, an unexpectedly high sensitivity to certain physical parameters is found. If this unexpected sensitivity has an important impact on the application of interest, a change in the design of the validation experiment may be needed, or indeed, a completely separate validation experiment may be needed. In addition, the limitations of the experimental facility should be directly factored into the design of the experiment. Examples of facility or diagnostic limitations are (a) an inability to obtain the range of parameters (e.g., load, temperature, velocity, time, radiation flux) needed to meet the goals of testing the physics models; (b) an inability to obtain the needed accuracy of measurements, including both SRQs and model input quantities; and (c) an inability to measure all of the input quantities (e.g., initial conditions, boundary conditions, material properties) needed for the code simulation.

The third strategy above refers to the importance of rigorously analyzing and quantitatively comparing the computational and experimental responses. As is shown in the top portion of Fig. 4, methods for quantitative comparison, i.e., validation metrics, have become an active topic of research (Dowding, 2001; Trucano et al., 2001, 2006; Hills and Trucano, 2002; Oberkampf and Trucano, 2002; Paez and Urbina, 2002; Hills and Leslie, 2003; Rutherford and Dowding, 2003; Chen et al., 2004; Dowding et al., 2004; Oberkampf and Barone, 2004, 2006; Mahadevan and Ramesh, 2005; Hills, 2006; Rebba et al., 2006). High quality validation metrics must use statistical procedures to compare the results of code calculations with the measurements of validation experiments. Because we stress that the overarching goal of validation experiments is to develop quantitative confidence so that the code can be used for its intended application, we have argued for the central role of validation metrics. Stated differently, we believe that predictive capability should be built directly on quantitatively assessed model accuracy, as opposed to making vague or ambiguous declarations that the model is "valid," or a foundation built on calibration of the model to all available data. The statistical inference literature provides a long history of statistical procedures that were developed for closely related inference tasks. Most of these procedures, however, yield probabilistic statements of "truth" or "falsehood," such as hypothesis testing, or the procedures are directed at the calibration of models, such as Bayesian updating. We believe it is important to refocus these procedures as much as possible on each of the three aspects of validation discussed in Fig. 4.

## 3. Recommendations for verification benchmarks

Section 3 presents our recommendations for constructing and using strong-sense benchmarks (SSBs) for code verification. These recommendations are directed toward improving the quality, accuracy, and documentation of existing benchmarks, as well as toward the development of new benchmarks. In the near term, these recommendations will likely be more valuable to computational analysts who have already developed some informal benchmarks. This audience would begin with an existing benchmark and follow the recommendations to develop the benchmark into an SSB. Importantly, an SSB should enable benchmark users and others to understand in detail the process that the benchmark developers followed to solve the benchmark problem. These recommendations would also be helpful to developers of new verification benchmarks in understanding the requirements of SSBs.

Our recommendations for verification benchmarks can be applied to many fields of physics and engineering and thus are not specific to any discipline. In Section 3.1, we discuss the features of constructing and also documenting a verification benchmark. Section 3.2 explains how to compare a code being tested (referred to as the candidate code) to the benchmark results. It is important to state here that Section 3 does

not address how to write the computer code for a verification benchmark.

### 3.1. Constructing verification benchmarks

High-quality verification benchmarks require both detailed documentation and exceptional procedures to ensure the accuracy of the computed results. The recommended documentation of a verification benchmark contains four elements (or parts): (1) conceptual description, (2) mathematical description, (3) accuracy assessment, and (4) additional user information. These parts are described in Sections 3.1.1–3.1.4, respectively.

#### 3.1.1. Conceptual description

The first part of the verification benchmark documentation is the conceptual description, i.e., information appropriate for the development of a conceptual model of the benchmark. The format of this description should be textual; no equations or symbols should be used. The reason for recommending that a textual description be given is that this format would be most usable in an electronic database of verification benchmarks that we believe should be constructed in the future. Our ideas about an electronic database are similar to those expressed by Rizzi and Vos (1998). With such a database, users could search for key words (provided in the textual benchmark descriptions) that would help them find benchmarks that might be applicable to particular problems of interest. The conceptual description should include five aspects of the verification benchmark, as discussed below. Note that the purpose of the benchmark is part of the fifth aspect.

The first aspect of the conceptual description should specify the general classes of physical processes being modeled in the benchmark. We refer to this aspect as the "title" of the benchmark. In fluid dynamics, for example, the description should give the general characteristics, such as steady or unsteady, class of fluid assumed (e.g., continuum or noncontinuum, viscous or inviscid, Newtonian or non-Newtonian, Reynolds-Averaged Navier-Stokes (RANS) equations or large eddy simulation (LES) or direct numerical simulation, compressible or incompressible, single phase or multiphase), spatial dimensionality, perfect gas, and all auxiliary models that are assumed (e.g., assumptions for a gas with vibrationally excited molecules; assumptions for chemically reacting gas; thermodynamic property assumptions; transport property assumptions; assumptions for chemical models, reactions, and rates; and turbulence model assumptions). In solid dynamics, for example, the description should include assumptions about equations of state, such as the choice of independent variables in tables; assumptions about solid behavior varying from elasticity to visco-plasticity; assumptions about material failure; and assumptions about the mixture behavior of complex nonhomogeneous materials. Note that the description should be given with respect to the classes of physics that are modeled in the benchmark, not the actual physics of interest in the particular application of interest.

The second aspect of the conceptual description should specify the initial conditions and boundary conditions exactly as they are characterized in the formulation of the conceptual model. Some examples in fluid dynamics are as follows: steady-state flow between parallel plates with infinite dimension in the plane of the plates, flow over a circular cylinder of infinite length with undisturbed flow at infinity, and flow over an impulsively started cube in an initially undisturbed flow. Some examples in solid dynamics are as follows: externally applied loads or damping, contact models, joint models, explosive loads or impulsive loads, and impact conditions (geometry and velocity). Included with the boundary conditions would be a statement of all the pertinent geometry dimensions or nondimensional parameters characterizing the problem (if any). Note that a statement of "far field" boundary conditions should clearly explain exactly what was used in the benchmark. For example, if the numerical solution benchmark imposed an undisturbed flow condition at some finite distance from an object in a fluid, then that condition should be carefully described. However, one could also impose an undisturbed flow condition at infinity using a coordinate transformation away from the object by mapping infinity to a finite point.

The third aspect of the conceptual description should specify various examples of important physical applications (or processes) to which the benchmark is relevant. Some examples in fluid dynamics are laminar wake flows, turbulent boundary layer separation over a smooth surface, impulsively started flows, laminar diffusion flames, shock/boundary layer separation, and natural convection in an enclosed space. Some examples in solid dynamics are linear structural response under impulsive loading, wave propagation excited by energy sources, explosive fragmentation, crater formation and evolution, and penetration events. The information in this aspect of the conceptual description will be particularly useful to individuals searching for benchmarks that are somewhat related to their actual application of interest.

The fourth aspect of the conceptual description should specify the type of benchmark. As discussed in Section 2.1.2, the benchmark type is one of the following: (1) a manufactured solution, (2) an analytical solution, (3) an ODE numerical solution, or (4) a PDE numerical solution. If the benchmark is type 1 or type 2, then the accuracy of the benchmark should allow the observed order of convergence of the candidate code to be computed. If the benchmark is type 3, or particularly type 4, it is questionable that the observed order of convergence can be computed for the candidate code because the accuracy of the numerical solutions from the benchmark may not be adequate. As a result, only an accuracy assessment of the SRQs of interest from the candidate solutions could be made by comparison with the benchmark solution.

The fifth aspect of the conceptual description should specify the numerical algorithms and/or code features that are being tested. Of the five aspects, this aspect reinforces the purpose of the verification benchmark by stipulating the algorithms being tested. Some examples of numerical algorithms that could be tested are as follows: the numerical method to capture a strong shock wave in three dimensions, the numerical method to determine whether it can accurately approximate specific types of discontinuities or singularities that occur either within the

solution domain or on the boundary, the numerical method to compute recontact during large plastic deformation of a structure, the numerical method to compute a denotation front in a granular mixture, and the numerical method to compute shock-induced phase transitions. The fifth aspect of the conceptual description should also specify whether the testing involves an isolated physics phenomenon or a type of physics coupling. In the latter case, for example, does the benchmark test the coupling of a shock wave and chemically reacting flow? or does the benchmark test the coupling of thermally-induced stresses in addition to mechanical stresses during large plastic deformation of a structure?

To better clarify how these five aspects would be applied in practice, we present conceptual descriptions, with their associated references, of four different types of benchmarks in fluid dynamics:

*Type 1 Benchmark Example* (*manufactured solution*)
(Ref. Eca et al., 2005; Eca and Hoekstra, 2006a,b)

*Title*: Steady, incompressible, turbulent flow, using one- and two-equation turbulence models for the RANS equations.
*Initial conditions and boundary conditions*: Boundary value problem, two-dimensional Cartesian coordinates, arbitrary boundary geometry, boundary conditions of the first, second, and third kind can be specified.
*Related physical processes*: Incompressible, internal or external turbulent flows, wall-bounded and free-shear-layer turbulent flows.
*Type of benchmark*: Manufactured solution.
*Numerical and/or code features tested*: Interaction of inertial, convective, and turbulence terms for RANS models.

*Type 2 Benchmark Example* (*analytical solution*)
(Ref. White, 1991)

*Title*: Unsteady, incompressible, laminar, Couette flow, using the Navier-Stokes equations.
*Initial conditions and boundary conditions*: Initial-boundary value problem, two-dimensional Cartesian coordinates, impulsive flow between flat plates where one plate instantaneously accelerates relative to a stationary plate with the fluid initially at rest.
*Related physical processes*: Impulsively started, laminar flows.
*Type of benchmark*: Analytical solution given by an infinite series.
*Numerical and/or code features tested*: Interaction of inertial and convective terms in one dimension; initial value singularity on one boundary at time zero.

*Type 3 Benchmark Example* (*ODE numerical solution*)
(Ref. White, 1991)

*Title*: Steady, incompressible, laminar flow of a boundary layer for a Newtonian fluid.
*Initial conditions and boundary conditions*: Initial-boundary value problem, in two-dimensional Cartesian coordinates, flow over a flat plate with zero pressure gradient.
*Related physical processes*: Attached, laminar boundary layer growth with no separation.
*Type of benchmark*: Blasius solution; numerical solution of a two-point boundary value problem.
*Numerical and/or code features tested*: Interaction of viscous and convective terms in a boundary layer attached to a flat surface.

*Type 4 Benchmark Example* (*PDE numerical solution*)
(Ref. Prabhakar and Reddy, 2006)

*Title*: Steady, incompressible, laminar flow using the Navier-Stokes equations.
*Initial conditions and boundary conditions*: Boundary value problem, two-dimensional Cartesian coordinates, flow inside a square cavity with one wall moving at constant speed (except near each moving wall corner), $R_l = 10^4$.
*Related physical processes*: Attached laminar flow with separation, laminar free-shear layer, flow with multiply induced vortices.
*Type of benchmark*: Numerical solution given by a finite element solution.
*Numerical and/or code features tested*: Interaction of viscous and convective terms in two dimensions; two points on the boundary that are nearly singular.

### 3.1.2. Mathematical description

The second part of the verification benchmark documentation is the mathematical description, i.e., a description of the mathematical model of the benchmark. The mathematical description should clearly and completely document the PDEs or ODES for the mathematical problem being solved. We want to stress here that the mathematical description of the benchmark must *not* include any feature of the discretization or numerical methods used to solve the PDEs and ODEs. Our recommendations for preparing the mathematical description are presented below.

(1) Clearly state all the assumptions used to formulate the mathematical problem description.
(2) Define all symbols used in the mathematical description of the benchmark, including any nondimensionalization used, and units of all dimensional quantities.
(3) State the PDEs, ODEs, or integral equations being solved, including all secondary models, or submodels. The mathematical statement of these models must be given in differential and/or integral form (i.e., continuum mathematics form), as opposed to the discretized form. Some examples of secondary models that could be given are equation-of-state models, thermodynamic models, transport property models, chemical reaction models, turbulence models, emissivity models, constitutive models for materials, material contact models, externally applied loads, opacity models, and neutron cross section models.
(4) If the solution is given by a manufactured solution, the source terms for the manufactured solution should be included in the documentation in two forms: (a) a traditional form for analytical equations; and (b) a form that is programmed in a commonly used programming language

such as C++ or FORTRAN. One should be able to electronically copy the programming for the source terms and insert it into a computer code, or into an input file for a code.

(5) Give a complete and unambiguous statement of all the initial conditions and boundary conditions used in the mathematical statement (i.e., item #3 above). The stated initial conditions and boundary conditions are those that are actually used for the solution to the PDEs and ODEs, *not* those that one would like to use in some practical application of the computational model. For example, if the benchmark solution is a numerical solution of a PDE (a type 4 benchmark), and the numerical solution uses an outflow boundary condition imposed at a finite distance from the flow region of interest, then that condition (in continuum mathematics form) should be given.

(6) State all of the SRQs of interest that are produced by the benchmark for comparison with the candidate code solution. The SRQs could be dependent variables in the mathematical model, functionals of dependent variables, or various types of probability measures of dependent variables or functionals. Examples of functionals are forces and moments acting on an object in a flow field, heat flux to a surface, location of a boundary-layer separation or reattachment line, and location of a vortex center. Functionals of interest should be stated in continuum mathematics form, not in discretized form. Examples of probability measures are probability density functions and cumulative distribution functions of the SRQs of interest.

(7) If any quantities provided in the description of the mathematical model are given by a random variable or are uncertain, provide a precise characterization of the quantity. For example, (a) if a quantity is given by a probability density function, then the family of distributions should be stated, along with all the parameters defining a specific distribution; and (b) if a quantity is given by an interval, e.g., no likelihood is specified over the interval, then the end points of the interval should be specified.

The overarching goal of this part of the verification benchmark documentation is to provide an unambiguous, reproducible mathematical characterization of the benchmark problem that eliminates all potential disagreement about what was mathematically intended in the mathematical model. We believe that this goal must be ruthlessly pursued and achieved. Any vagueness, ambiguity, or missing detail in the mathematical model must be replaced with explicit specification.

A comment should be made here about the practice of incorporating numerical approximations or features directly into the mathematical models of the physics. An example in fluid dynamics is seen in many LES models of turbulence. Many researchers who solve the LES equations will define the length scale of turbulence to be modeled as that determined by the local discretization scale used in the numerical simulation. That is, the subgrid turbulence scale is defined to be all spatial scales smaller than the local mesh that the researchers happen to be using. An example in fracture dynamics is seen in the modeling of crack propagation through a material. Some researchers, but thankfully fewer in recent times, will define the spatial scale of the crack tip radius to be the same as the local mesh resolution used in a particular numerical solution.

We strongly argue against the practice of connecting physical modeling scales, either spatial or temporal, with numerical discretization scales. Our arguments are particularly compelling when verification benchmarks are being solved. The rationale for our objection is twofold. First, combining physics modeling with numerical approximations intertwines two very different issues. Models of physics should be stated in a way that does not, in any way, depend on how the numerical solution is obtained. Mathematical models of physics should depend only on the spatial and temporal scales in the physics being modeled. Second, if a physics model is defined to be dependent on numerical solution approximations, then changes in the numerical approximations, e.g., mesh resolution, will result in changes in the physics model. Suppose one wanted to use a different class of numerical methods to solve the mathematical model, such as a higher-order method, then, even with the same mesh resolution, two different numerical solutions would exist; neither solution would have any meaning with respect to the differential equations stated in the mathematical model. Mixing physics modeling and numerical solution approximations is, in our view, as bad as mixing different dimensional units—it makes no sense. Physics modeling scales, typically dimensional scales in length or time, should be based on the physical scales that are captured in the differential equations of the mathematical model.

#### *3.1.3. Accuracy assessment*

In this part of the verification benchmark documentation, the numerical accuracy of the benchmark should be critically assessed, and the means of assessment should be carefully described. Thus, preparing this part of the benchmark documentation is dependent on having executed the code to solve the benchmark problem. The assessment procedure and the accuracy assessment result should be described for each SRQ that is provided by the benchmark. The accuracy assessment should be provided, if appropriate, as a function of (a) spatial coordinates; (b) temporal coordinate; and (c) parameters provided in the mathematical model, e.g., Reynolds number, Mach number, externally applied load, heat flux, and boundary condition parameter. In general, the accuracy assessment of the SRQs depends on all the independent variables and parameters in the model. The purpose of this assessment is to provide a definitive pedigree for the benchmark accuracy that is unambiguous and objective. This task clearly becomes much more difficult as we progress from a type 1 benchmark to a type 4 benchmark. False pedigrees often lie at the heart of failed, complex benchmark efforts centered on PDE numerical solutions. Many managers and organizations are fond of complex, high-visibility benchmarks, but the credibility of these benchmarks invariably disappears when the details of the benchmarks are examined.

The accuracy of a benchmark will greatly depend on the type of benchmark solution that has been computed. We now discuss particular accuracy assessment issues that are unique to each type of benchmark.

*3.1.3.1. Accuracy assessment issues for type 1 benchmark (manufactured solutions).* Manufactured solutions are all composed of well-known, elementary functions, such as circular functions and exponential functions. The accuracy issue in manufactured solutions centers on the accuracy, or correctness, of all the source terms that are derived and then placed on the right-hand side of the PDE. The two texts (Roache, 1998a; Knupp and Salari, 2002) that deal with the MMS recommend a number of practices and procedures that are very helpful in using the method. A few of these recommendations are included here: (1) Do not try to derive the source terms by hand. Use symbolic manipulation software such as Mathematica® or MATLAB® to derive the source terms. (2) When the source terms are derived, do not try to program them by hand. The suggested practice is to electronically copy the terms from the symbolic manipulator output directly into the software solving the PDEs. (3) To check the correctness of the output from the symbolic manipulation software, we recommend the use of two different software packages. (4) When selecting a manufactured solution form and its associated free parameters, choose the solution form and the parameters so that when the solution is substituted into the original PDE, all the terms in the original PDE are reasonably balanced in magnitude. This balance aids in the identification of terms that contain a programming error.

*3.1.3.2. Accuracy assessment issues for type 2 benchmark (analytical solution).* If the benchmark solution is given in terms of a closed-form solution, the accuracy is usually near machine precision. As used here, a "closed-form solution" is a solution that can be expressed analytically in terms of a bounded number of well-known functions. We also presume that the derivation of the solution can be fully comprehended by knowledgeable people who use it as a benchmark. If the derivation is incomplete or otherwise not fully available for critical scrutiny, it is unlikely that the benchmark will be credible. If the benchmark is not a closed-form solution, then one must very carefully estimate the accuracy of the solution. If the analytical solution is given by an infinite series, then the accuracy is determined by the rate of convergence and the number of terms that are included before the sequence is truncated. One cannot, in general, estimate the accuracy of an analytical solution given by an infinite series by simply comparing how much the solution changes when one more term in the infinite series has been added. If the analytical solution contains an integral, or an iterative solution of an algebraic or transcendental equation, one must estimate the numerical error involved. For example, in the Type 2 Benchmark Example given in Section 3.1.1, the solution for the unsteady Couette flow is given by an infinite series. The convergence rate of the series drastically depends on the time chosen. For times near zero, the convergence rate is extremely poor compared to large times because of the existence of the singularity when the time equals zero, i.e., the start of a simulation.

*3.1.3.3. Accuracy assessment issues for type 3 benchmark (ODE numerical solution).* Benchmark solutions obtained by the numerical solution to a set of ODEs can be initial value problems or boundary value problems. The accuracy of solutions to these problems primarily depends on the sophistication and reliability of the numerical integrator used to compute the solutions. For benchmark solutions, it is recommended that a high-order accuracy integration technique be used, along with a variable step-size procedure that is adjusted according to a per-step, relative-error criteria specified by the user. If possible, two different numerical integrators should be used and the results compared. It is recommended that the order of convergence of the ODE integrator be higher than the formal order of convergence of the candidate solution being tested. If a fixed-order accuracy method is employed, then Richardson extrapolation can be used to estimate the error of the numerical solution for each SRQ of interest. An example of an efficient, high-order accuracy procedure is an embedded Runge-Kutta method of order 6 or 7. Additional complexity, and inaccuracy, is introduced if one numerically solves a boundary value problem. Solutions to boundary value problems should include user-specified control of the error along both boundaries. If a singularity exists along any boundary, or as an initial condition, then methods must be developed to estimate how the numerical error near the singularity propagates into the solution domain.

*3.1.3.4. Accuracy assessment issues for type 4 benchmark (PDE numerical solution).* Benchmark solutions obtained by the numerical solution of a set of PDEs present the most challenging accuracy assessment issues. Compared to the first three types of benchmarks, type 4 benchmarks require that much more detail be provided. Our recommendations for conducting and documenting the accuracy assessment for a type 4 benchmark are presented below. Importantly, the information provided should enable someone both to understand the estimated accuracy of the benchmark and to evaluate the strength of the procedure used to estimate the accuracy.

(1) Describe all the iterative procedures and convergence criteria used in all aspects of the numerical solution, e.g., the iterative procedure and convergence criteria for iterative solution of a nonlinear boundary value problem, the iterative procedure and convergence criteria for intra-time-step iterations.
(2) Compute a series of solutions using at least three different mesh resolutions, and use Richardson extrapolation to estimate the numerical error over the entire solution domain for each of the SRQs of interest. Also, using the multiple mesh-resolution results, estimate the observed order of convergence of the solution for each SRQ, and compare it with the formal order of convergence expected from the method. It could be argued that some of the *a posteriori* finite element error estimation procedures, such as recovery methods or residual methods, could be used instead of Richardson extrapolation (Ainsworth and Oden, 2000; Babuska and Strouboulis, 2001). We should note, however, that there are some practical difficulties with most of these methods. First, some methods provide global error norms rather than error estimates on the SRQs of interest, such as error estimates of local dependent variables. Second, some methods only provide error estimates to within some unknown con-

stant. Third, very few methods have been developed for nonlinear parabolic and hyperbolic PDEs. Fourth, any substantial change to the PDE or any submodel requires that the error estimation equation be derived again. And fifth, it is poorly understood at present how *a posteriori* finite element error estimators are affected by the lack of continuity of higher derivatives of dependent variables and by singularities. Experience has shown that Richardson extrapolation is more robust than *a posteriori* finite element error estimators, probably because Richardson extrapolation is directly based on a power-series expansion of the SRQ of interest.

(3) If the benchmark problem is an initial value problem, compute a sequence of solutions using at least three different temporal resolutions, and use Richardson extrapolation to estimate the numerical error over the entire solution domain for each of the SRQs of interest. Also, using the multiple solutions, estimate the observed order of temporal accuracy and compare it with the formal order of temporal accuracy for each SRQ. In estimating the temporal accuracy, include the coupling of the temporal and spatial accuracy in the Richardson extrapolation equations.

(4) If a singularity exists inside the solution domain or on any boundary, or in the initial conditions, provide strong evidence that the numerical solution is not contaminated by error propagated away from the singularity. One method that adds credence to a numerical solution with a singularity is to use two markedly different numerical methods to solve the same problem and then show the results from both methods for all SRQs of interest. Though technically demanding, a preferable approach for dealing with a singularity is to analytically eliminate the singularity from the problem in some fashion. The Type 4 Benchmark Example given in Section 3.1.1, the driven-cavity problem, demonstrates some of the difficulties encountered with solutions containing singularities. Prabhakar and Reddy (2006) eliminated the two singularities in the moving-lid corners by replacing the fixed speed of the moving lid with a speed that varies spatially near each of the corners. They clearly state that had they not removed the singularities, their numerical procedure would not have converged. We are not aware of any solutions to the driven-cavity problem published prior to Prabhakar and Reddy's work that removed the singularities in the corners. As a result, we are highly suspicious of the accuracy of all earlier numerical solutions to the driven-cavity problem.

### 3.1.4. Additional user information

The fourth part of the verification benchmark documentation should include additional information that would be helpful to users of the benchmarks. For example, such information might assist a researcher in investigating how the accuracy of a benchmark could be improved or how the generality of the benchmark could be extended. Similarly, if a user's candidate solution did not satisfactorily compare with the benchmark, some small documented detail might help the user discover the cause of the discrepancy.

Several pieces of information should be provided in this part of the documentation, regardless of the type of benchmark computed. Appropriate descriptions of the following should be given: (a) computer hardware used; (b) operating system and version; (c) compiler type and version and any pertinent compiler options used; (d) arithmetic precision; (e) programming language used in the source code; (f) what type and how extensive have been the code verification activities; (g) computer run time for each of the solutions documented in the benchmark; and, of course; (h) authorship of the benchmark results and their affiliated organization. Some of the additional information that should be included differs significantly for each type of benchmark. We now discuss particular information needs that are unique to each type of benchmark and also provide recommendations for addressing these needs.

*3.1.4.1. Unique information needs for type 1 benchmark (manufactured solution).* The symbolic manipulation software used to derive the source terms should be stated, along with the version number of the software. If two different symbolic manipulation software packages are used to serve as a check, then this should be stated. If this is done, one should be certain that each package is unrelated to the other. For example, the symbolic manipulation kernel in MATLAB® from The MathWorks is the same as the symbolic manipulation kernel in Maple™ from Maplesoft.

*3.1.4.2. Unique information needs for type 2 benchmark (analytical solution).* The analytical solution should be documented in the traditional form of equations and explanatory text. If the benchmark solution is given by an infinite series, a description should be given of the method used to estimate the error due to truncation of the series. If all terms in the series have the same sign, then one method for estimating this error is to compute a curve fit of the magnitude of each term as a function of the number of the term in the series. If the terms have alternating signs, then a curve fit of the magnitude of the sum of pairs of terms can be computed. With a proper choice of functional form, the curve fit can then be extrapolated to infinity. Then the sum of the truncated terms can be computed to estimate the error due to the truncated series. If the benchmark solution is given by an integral or by an iterative solution of an algebraic or transcendental equation, the numerical method used to compute the integral and the iterative solution should be given. Adequate references must be provided for the analytical solution, along with its derivation, if possible. The references should be publicly available.

*3.1.4.3. Unique information needs for type 3 benchmark (ODE numerical solution).* A detailed description should be provided of the numerical method used to solve the ODE. If the numerical integrator is contained in a software package, give (a) a description and version number of the package and (b) information concerning what type of code verification has been documented on the package. If possible, the software package should be included in the additional user information portion of the benchmark. Also, if any tabular data is used in any mathematical submodel, then all of the numerical data should be provided, along with a description of the interpolation procedure used for the tabular data.

*3.1.4.4. Unique information needs for type 4 benchmark (PDE numerical solution).* A detailed description should be provided of all of the numerical methods used in all aspects of the solution procedure. Our recommendations for preparing this information include the following:

- Describe all of the numerical algorithms used to discretize the PDEs and all submodels, including any parameters or constants that might be associated with the numerical algorithms, e.g., artificial damping parameters, and smoothing parameters.
- If the geometry contains any complex features, describe in detail the geometry and explain how it was computed, e.g., any interpolation procedures used to construct the geometry.
- Describe how the spatial mesh was generated, especially all the clustering features of the mesh, and provide the coordinates of all mesh elements.
- Describe how all the sequences of meshes with different levels of mesh refinement are related to one another. For example, were the multiple meshes generated by starting with the finest mesh and then coarsening? or was the process done in the reverse order?
- State the formal order of convergence of all the numerical methods used to solve the PDEs, including numerically computed Jacobians in mapping the physical space to the computational space, and any numerical processing procedures, such as interpolation, integration, or differencing, that were used to compute the SRQs of interest.
- Describe the computer code, including its version number, and state whether the code is available for public dissemination.

### *3.2. Comparing candidate code results with verification benchmarks*

As discussed in Section 1, we are only interested in comparisons of a candidate code with a benchmark for the purpose of assessing the accuracy of the results of the candidate code. Issues related to computing-speed performance or to robustness of the candidate code are not of particular interest here. Given this context, how one would want to report the results obtained from comparing a candidate solution to a benchmark solution depends on the purpose of making the comparison. Suppose the purpose of the comparison is similar to one of the following: (a) making a preliminary assessment of the accuracy of a code that is in development, (b) investigating the accuracy of a new numerical algorithm implemented in a code, or (c) conducting a proprietary investigation of the accuracy of a code that is in competition with your own commercial code. We would characterize all these types of comparisons as "informal," given that the results of the comparison are for restricted or preliminary use, and they may not be documented.

In this paper, we are interested in "formal" comparisons of candidate results and benchmark results. Some examples of the use of formal comparisons are as follows: (a) a potential software customer wants to compare the accuracy obtained from competing commercial codes; (b) a large organization that develops its own codes for internal use for high-consequence systems wants to determine how its codes compare with industry-standard benchmarks; (c) a governmental regulatory organization wants to require that certain verification benchmarks be passed before a code could be used for performing analyses of high-consequence systems; (d) an accident investigation committee wants to determine whether there were any deficiencies in the software that was used to analyze the performance and safety of a system that failed; and (e) a commercial software company wants to use the results of formal comparisons of its code with benchmarks in its marketing program.

Even though our interest is in rigorous comparisons, we believe that these comparisons should *not* be included in the benchmark database. Our viewpoint is contrary to the views expressed by Rizzi and Vos (1998) and Vos et al. (2002). However, one must recognize that the database these researchers have envisioned and the databases that have been constructed in Europe are developed with a weaker form of benchmark than the benchmarks we are proposing in this paper. Rizzi and Vos, and Vos et al. believe that comparison results that have been obtained should be included in the database if the individuals who computed the results choose to put them in the database. It is our view that *if* the benchmarks in the database are indeed *SSBs*, such executed comparisons add nothing to the database. If the candidate code results have met all the stringent requirements for inclusion in the database, the candidate results could be included as a new benchmark for the same problem. Alternatively, the new solution could possibly replace the existing benchmark if it has a stronger pedigree than the existing benchmark. As discussed in Section 5, there must be a well-defined and formal review process for deciding which solutions can be included in the SSB database.

Thus, we advocate that separate documentation be prepared to describe the formal comparisons. The formats in which the documentation is prepared and presented (e.g., reports, graphs, etc.) should be determined by the developers of the documentation, based on the purpose for which the results of a code's performance is being published. To achieve some of the goals suggested for formal comparisons, the documentation of the comparisons should contain much of the same information described previously in Sections 3.1.1–3.1.4. The key piece of information that is of interest in the documentation is, did the candidate code pass the benchmark? The most common method of answering this question is by comparing a computed result for an SRQ from a candidate code with the comparable result from an SSB. Although this comparison is useful, it has two significant disadvantages. First, the accuracy requirement for comparing the candidate and benchmark SRQs is quite arbitrary. For example, Should one require an accuracy of 1% or an accuracy of 0.1% or machine precision accuracy when comparing results? Saying that the accuracy required depends on the application of interest defeats the purpose of the benchmark. Second, the accuracy of the candidate result will depend directly on the mesh and the temporal resolution that are used in the computed result. That is, the candidate result will depend in a continuous manner on both the mesh and the temporal resolution. As discussed in Section 2.1.2, the most definitive test of the accuracy of a code is to determine the observed order of convergence.

For type 1 and type 2 benchmarks, the accuracy of the benchmarks should be adequate to determine the observed order of convergence by using the benchmark and solutions from three different mesh resolutions of the candidate code. For a type 3 benchmark, this approach may not be possible because the accuracy of the benchmark may not be adequate. For a type 4 benchmark, it is likely that the accuracy of the benchmark will not be adequate to reliably determine the observed order of convergence of the candidate code. As a result, different measures of "pass" and "fail" must be assigned to each type of benchmark with which the candidate code is being compared.

When presenting the observed order of convergence for the candidate code, there are two criteria one might use to determine the assessed order of convergence of the candidate code, i.e., the pass/fail status of the candidate code compared to the benchmark. First, one may choose to require that the observed order of convergence of the candidate match its stated formal order of convergence. Or second, one may choose the much weaker criterion that the observed order of convergence of the candidate code be positive, i.e., the minimum requirement that it converged to the correct answer. We believe, however, if the observed order of convergence is close to zero, then it is unlikely that the candidate code is correct. Regardless of which criterion is chosen, the observed order of convergence should be reported in the documentation as a plot of the observed order of convergence as a function of mesh and/or temporal resolution. In such a plot, one can discern the observed order of convergence in the asymptotic region for the particular SRQ.

If the observed order of convergence cannot be computed for the candidate code, then one is left with simply comparing the candidate result for an SRQ with the corresponding benchmark result. If this comparison is used, it is recommended that the results be shown as a difference between the candidate code and the benchmark as a function of mesh and/or temporal resolution. If the candidate code is capable of computing the solution as accurately as the benchmark, then the difference plotted would start to show erratic results for fine-mesh resolutions.

## 4. Recommendations for validation benchmarks

In Section 2.2.2, we briefly discussed our views on the unique characteristics of validation experiments. As pointed out, a validation experiment is more than a traditional, high-quality experiment. It must provide information that is typically not available in traditional experiments, and it is optimized for nontraditional customers, such as mathematical model builders and simulation analysts. Since most traditional experiments available in the published literature have not been designed as validation experiments, some of the recommended characteristics to be discussed for strong-sense benchmarks (SSBs) will seem rather idealistic and impractical to obtain. However, as new experiments are conducted in the future, these recommendations could be used for the design and acquisition of new high-quality validation benchmarks.

High-quality validation benchmarks will be much more feasible to obtain at the lower tiers of the validation hierarchy discussed earlier. As one proceeds to higher tiers, i.e., more complex physical systems, in the hierarchy, the number and importance of the unmeasured input quantities will decrease the ability to critically assess the computational model of interest. Stated differently, comparing experimental data obtained from complex systems with computational results inevitably becomes a process of calibrating the very large number of either unmeasured or poorly known parameters in the models. Thus, most of the recommendations for validation benchmarks in Section 4.1 deal with the common theme: measurement and documentation by the experimentalist of essentially all input quantities needed in the code to minimize the degree of calibration of the physics modeling parameters by the computational analyst. In Section 4.2, the same theme is addressed, but there it is oriented toward the computational analyst who is conducting the comparison of the candidate code with the validation benchmark.

### 4.1. Constructing validation benchmarks

The activity of constructing and documenting validation benchmarks is primarily the responsibility of the experimentalist. As discussed with respect to Fig. 4, validation benchmarks are intended to address the issue of model accuracy assessment. Issues related to the accuracy requirements for a particular application, or the accuracy of the model when it is extrapolated to other intended uses, are not addressed in this discussion on constructing validation benchmarks. Furthermore, issues pertaining to code verification, solution verification, and modeling assumptions are not dealt with in this section, as those issues are properly addressed in Section 4.2. As we have emphasized, there is a logical dependence of the quality of validation upon code and solution verification.

High-quality validation benchmarks require both detailed documentation and exceptional procedures to ensure high accuracy of the benchmarks. The recommended documentation of a validation benchmark contains four elements (or parts): (1) conceptual description; (2) experimental description; (3) uncertainty quantification (UQ) of benchmark measurements; and (4) additional user information. These parts are described in Sections 4.1.1–4.1.4, respectively.

To clarify some of the recommendations, we give an example of a hypothetical benchmark experiment in fluid dynamics. This example is carried through the discussion of each of the following subsections. Not every detailed piece of experimental information needed for the benchmark is discussed in this example, but we highlight those elements of our experiment that are not commonly included in the execution and documentation of an experiment.

#### 4.1.1. Conceptual description

The first part of the validation benchmark documentation is the conceptual description, i.e., information appropriate for the development of a conceptual model of the benchmark. The format of this description should be textual; no equations or symbols should be used. The reason for recommending that a textual description be given is that this format would be most usable in an electronic database of validation benchmarks that we believe should be constructed in the future. The conceptual

description should include three elements, namely, the primary types of physics being tested, the SRQs measured in the experiment, and related engineering applications. Listed below are our recommendations for developing a conceptual description for these important elements.

(1) Describe the primary types of physics, or coupled physics, that the benchmark is intended to test in the computational modeling. If appropriate, a description should be given that is divided into two categories denoting the importance of the physics being tested: the primary physical processes occurring in the experiment, and the secondary physical processes occurring in the experiment. This categorization will assist both computational analysts and developers of physics models in searching the validation database for experiments that are aligned with their immediate interests. In designing validation experiments, one should maximize the effect of the physics of interest and minimize the effects of all other physical processes not of interest. Our example in fluid dynamics begins with the following: *primary physics occurring* – incompressible, turbulent flow with large separated regions over a circular cylinder with heat transfer; *Secondary physics occurring* – small effect of variable thermodynamic and transport properties near a heated surface and in a wake region.
(2) List both the quantitative and qualitative SRQs measured in the experiment. The quantitative SRQs could be steady-state, time-averaged or frequency-averaged, time-resolved or frequency-resolved measurements. We have found that qualitative measurements, such as video imaging of the physics phenomena during the experiment, can be very useful in guiding the computational analyst in the appropriate assumptions that should be made for modeling the experiment and also for aiding the experimentalist in diagnosing any unforeseen problems with the experiment. Continuing with our example in fluid dynamics, the SRQ listing is as follows: *system responses quantitatively measured* – three-dimensional, unsteady velocity measurements in streamwise planes normal to the cylinder, and high-frequency, surface pressure measurements in the wake of the cylinder, system responses qualitatively measured – flow-field visualization provided by marker-dye injection, high-speed-digital-video imaging of the flow field.
(3) Describe some of the important engineering applications at higher levels in a validation hierarchy to which this benchmark could be related. Since complex engineering systems, or subsystems, of interest occur at higher tiers in the validation hierarchy, some examples should be provided so that electronic searches of the validation database could find benchmarks that may be of interest to a wide range of applications. For our fluid dynamics example, the following engineering applications are provided: *related applications of interest* – flow inside heat exchangers, flow across tube-bundles, natural convection inside cavities, liquid cooling of internal combustion engines, forced and natural convection over circuit boards.

*4.1.2. Experimental description*

The second part of the validation benchmark documentation is the experimental description. The description should provide a wide variety of necessary detailed information about the geometry of the experiment; the boundary conditions, initial conditions, and auxiliary data; and the SRQs measured. Specification of these elements should include, as applicable, input data needed for the computer code, measurement techniques used in the experiment, any data reduction and processing techniques required, and details about the experimental facility. A suggested approach to preparing the experimental description is presented below.

(1) Describe the geometry of the experiment conducted, along with any supplementary geometry experiments that were conducted in support of the benchmark experiment. A supplementary geometry experiment is one that could be simulated by the computational scientist with much higher accuracy and confidence than the primary geometry of interest. For our fluid dynamics example, we have the following geometry-related information: *Geometry* – flow over a circular cylinder near a flat, solid wall in a water tunnel. The cylinder was mounted at various distances from the wall: 0.0, 0.1, 0.2, and 0.5 diameters.
(2) Supplementary geometry—flow inside the water tunnel without the cylinder in the test section Specify all the measured boundary conditions, initial conditions, material properties, imperfections in the test geometry or experimental facility, forcing functions, surface properties, transport properties, thermodynamic properties, mass properties, etc. The specification of boundary conditions should include computer-aided-design (CAD) files of the exact geometry that was used in the experiment and should be presented in a commonly used format. If appropriate, two types of CAD files should be provided: (a) a file of the geometry as it was manufactured and assembled; and (b) a file of the geometry as it existed during the experiment, e.g., under thermal or mechanical loading. The CAD files can significantly diminish the possible misinterpretation or ambiguity present in traditional design drawings and greatly reduce the time required by computational scientists to construct a mesh of the geometry. Validation experiments should be designed to minimize the complexities and difficulties with which computational analysts must deal, *if* these problems are not important to assessment of the physics models of interest. For our fluid dynamics example, we have the following information about the boundary conditions: *Boundary conditions* – a solid circular cylinder was heated over its entire length using electrical-resistance heating. The cylinder was mounted near the bottom wall of a water tunnel, and it spanned the entire width of the test section. Over the length of the test section, the tunnel had a square cross section 10 cm × 10 cm. The diameter of the cylinder was 1 cm, and it was placed 20 cm from the beginning of the test section. The test section was 100 cm long. All the tunnel walls had a turbulent boundary layer approaching the test section. The

three-dimensional, unsteady velocity field was measured over the entire inflow plane at the beginning of the test section for each Reynolds number tested. Increased spatial resolution of the velocity field was attained near each of the tunnel walls. The water temperature was measured at the beginning of the test section and as a function of time during the experiment. The water was de-aerated to eliminate bubbles. Measurements were made for two Reynolds numbers: $10 \times 10^3$ and $100 \times 10^3$. These numbers were based on average inflow velocity, kinematic viscosity of the water, and diameter of the cylinder. Time-averaged static pressure measurements were made in the middle of each tunnel wall at three locations: at the beginning, middle, and end of the test section. The heat flux per unit length along the cylinder was measured. The heat flux leaking from the ends of the cylinder was measured. For 100 cm past the end of the test section, each wall of the water tunnel was set at the same diverging angle of 5°, resulting in an increasing cross-sectional area. Note that accompanying the above textual description would be detailed drawings and a CAD file of the geometry of interest and the water tunnel, as well as measurement locations for the boundary conditions.

(3) Specify all the SRQs that were both quantitatively and qualitatively measured, along with a detailed description of the diagnostic techniques, analog-to-digital sampling, signal filtering, signal conditioning, and time- or frequency-averaging methods. For our fluid dynamics example, we have the following SRQ specifications: system responses quantitatively measured—three-dimensional, unsteady velocity measurements in three planes normal to the cylinder. One plane was in the middle of the cylinder. The other two planes were halfway between the middle of the cylinder and each side wall. The planes extended from 5 diameters upstream of the cylinder to 10 diameters downstream of the cylinder, and from the lower to the upper wall of the test section. Velocity measurements were made using particle imaging velocimetry in a rectangular grid pattern at 5000 points in each plane. Velocity measurements were made at a frequency of 1/s for a time period of 1000 s. Time-averaged velocity measurements are also available over the 1000 s period. High-frequency surface pressure measurements were made on the wall of the tunnel at 0, 1, and 5 diameters downstream of the cylinder; *system responses qualitatively measured*—marker dye was injected along a narrow slit in the wall near the cylinder and parallel to the cylinder at a location of 5 diameters upstream of the cylinder. Digital video images were recorded of each experiment at a framing rate of 100/s. The unsteady cellular structure in the wake of the cylinder can be seen at each Reynolds number tested, along with the change in wake structure near the sidewalls of the test section.

#### *4.1.3. Uncertainty quantification of benchmark measurements*

The third part of the validation benchmark documentation should provide estimates of experimental uncertainty for all the SRQs measured, as well as uncertainty estimates of all the quantities that could be used as possible inputs for the computational simulation, such as boundary conditions, initial conditions, material properties, geometrical features, etc. A suggested approach to estimating and documenting the UQ is presented below.

(1) Describe all the instrument, diagnostic, and facility calibration procedures. Particular emphasis in calibration procedures should be placed on identifying, and possibly estimating, subtle bias errors in calibrations, e.g., shifts in diagnostic measurements due to temperature, pressure, time, reference frequencies, and so forth. In designing validation experiments, one should attempt to use multiple diagnostic techniques to measure both SRQs and input quantities. By comparing results from multiple measurement techniques, one can better identify possible bias (systematic) errors in the measurements. For our fluid dynamics example, an experimentalist could use different diagnostic techniques to identify bias errors in the optical calibration of particle imaging velocimetry measurements. The experimentalist could also use different techniques to determine possible temperature bias effects on the high-frequency surface pressure measurements aft of the cylinder.

(2) Describe whether an input quantity needed for the computational simulation is a controlled quantity or an uncontrolled quantity in the experiment. A controlled quantity is one that can be adjusted, to a large degree, by the experimentalist or by procedures related to the operation of the experimental facility. An uncontrolled quantity is one over which the experimentalist has little or no control, such as atmospheric weather conditions, impact location of an object on an irregular surface, turbulence spectrum and spatial variability in a wind tunnel, and unit-to-unit variability of material samples. If a quantity is uncontrolled but can be measured, e.g., atmospheric weather conditions, then uncertainty in the measurement should be given. If the quantity is an uncontrolled quantity but was randomly drawn from a population, then the population should be well characterized before the experiment. For example, if material testing is being conducted on a number of small specimens (coupons), the needed input material properties should be characterized by a probability distribution that was constructed by a large number of random draws from the sample population. There are also situations where there are a very limited number of specimens and the specimens are destroyed in the characterization process. In such cases, large uncertainty exists in the characterization of the population, resulting in an ensemble of possible probability distributions. This large uncertainty damages the quality of the validation benchmark, but it is sometimes unavoidable because of cost considerations. Alternately, the characterization of the specimen population would occur during the validation process via a calibration activity. This latter approach, although less desirable because it combines validation and calibration, is sometimes unavoidable.

(3) Provide estimates of both the bias error and the random (precision) error of the quantities measured. The uncertainty in measured quantities could be characterized as one of the following: an interval (i.e., there is a single true value that is believed to lie in the stated interval, but no other information is available concerning the true value); an imprecise probability distribution (i.e., the true quantity is a random variable characterized by a known family of probability distributions, but the parameters of the probability distribution are only stated as intervals); and a precise probability distribution (i.e., the true quantity is a random variable characterized by a probability distribution with accurately known parameters). It has been found that one of the most effective methods of quantifying experimental uncertainties, particularly bias errors, is to conduct the same experiment in multiple experimental facilities, preferably using different diagnostic techniques. The time and cost involved in conducting experiments at multiple facilities will commonly cause most project managers and funding sources to lose interest.

(4) Describe and justify the UQ procedures that were used for each measured quantity. Some examples of UQ procedures, from least desirable to most desirable, follow: experience of the experimentalist in previous experiments using similar techniques in the same facility; measurement of some of the components contributing to uncertainty, but no formal procedure for estimating uncertainty; propagation of contributing uncertainties to formally estimate uncertainty in an SRQ (Coleman and Steele, 1999); and statistical design-of-experiment procedures to directly estimate the uncertainty in SRQs using multiple realizations of the experimental measurements under varying conditions (Oberkampf et al., 1995; Aeschliman and Oberkampf, 1998; Oberkampf and Blottner, 1998; Montgomery, 2000; Box et al., 2005). This last procedure, usually referred to as a statistical blocking procedure, can quantify certain types of correlated-bias errors, such as those due to nonuniformity in the flow field of a wind tunnel, imperfections in the model used in a wind tunnel experiment, certain types of misalignment in a load cell, and asymmetries in the thermal heating of components.

#### 4.1.4. Additional user information

The fourth part of the validation benchmark documentation should include all the traditional documentation associated with archiving high-quality experiments. In addition, the documentation should include details that could possibly assist users of the benchmark in several ways. First, information on the experimental technique, experimental procedures, experimental facility, boundary condition, initial conditions, etc., should be provided that could help the computational modeler choose different modeling assumptions than the experimentalist might have thought the modeler would have used. For example, the modeler may choose to use a three-dimensional Cartesian coordinate system instead of a two-dimensional axisymmetric coordinate system, or the modeler may want to include the actual nonuniformities in either the component tested or the facility being used in the experiment. Second, information should be provided on the experimental operating procedures. Based on this description, the modeler may choose to represent experimentally reported measurement uncertainties differently than what might be expected. Third, another experimentalist may choose to conduct the same experiment in their facility and submit the results either to supplement the existing benchmark or to possibly replace it.

### 4.2. Comparing candidate code results with validation benchmarks

The activity of comparing candidate code results with validation benchmarks and preparing documentation for this activity is the responsibility of the computational analyst. As discussed in Section 3.2, we are only interested in formal comparisons of code results with validation benchmarks. Also, as explained previously, the code results and comparisons with the validation benchmarks should *not* be included in the database. As with the comparison of candidate code results with verification benchmarks, the formats in which the documentation is prepared and presented (e.g., reports, graphs, etc.) would be determined by the developers of the documentation, based on the purpose for which the results of a code's performance is being published.

In the comparison of code results with validation data, we do not believe there is an acceptable way, in general, to answer the question, did the code pass the validation benchmark? Our viewpoint can be explained from two perspectives. First, we view the assessment of model accuracy by comparison with experimental data as a "continuum" in the sense of the validation metrics discussed in Section 2.2.1. We believe that validation metrics are the fundamental operators in assessing model accuracy. A validation metric is a difference operator that can yield a deterministic result, a precise probability distribution, or an imprecise probability distribution, with each, preferably, having some type of associated confidence measure. Stated differently, validation metrics are simply measures of agreement between simulations and experiments that have no fundamental "good" or "bad" associated with them. Second, to state that a benchmark has passed, one would have to have some stated accuracy requirement for an application of interest, as addressed in the discussion of Fig. 4. The accuracy requirement should, we believe, be determined by the application of interest, not by some vague concept related to the philosophy of science or, for example, related to the amount of scatter that exists in the experimental data. In addition, validation metrics can be applied to several different SRQs from a validation benchmark. It is expected that the metric results for some of the SRQs will meet accuracy requirements, and some will not. Then, as we have observed in real engineering projects, additional discussions will ensue with regard to the appropriateness of the accuracy requirements, as well as the cost, schedule, and performance of the engineering system of interest. The consequence of our viewpoint is that the comparison of code results with validation benchmarks should be formally documented, but no pass or fail assignment should be given.

The types of information that should be included in the documentation regarding the comparison of code results with

validation benchmarks is a combination of (a) the recommended documentation described previously in Sections 3.1 and 3.2 for constructing and comparing with verification benchmarks, especially for a type 4 benchmark; and (b) the recommended documentation provided in Section 4.1.1 for constructing validation benchmarks. As discussed below, most of the recommendations for comparing candidate code results with validation benchmarks will stress the common theme: exposing and explaining in the documentation of the computational analysis any "tuning" of physics modeling choices or numerical parameters that has been done to improve comparisons with the experimental data. As is well known, there is a great deal of flexibility in computational simulations to tune methods and parameters to obtain good agreement with known experimental measurements. Left unfettered or hidden, this flexibility greatly diminishes the value of documenting comparisons of computational simulations with already known measurements. And, as is widely recognized, the most valuable validation benchmarks are those for blind comparisons, i.e., comparisons with experimental measurements that are *not known* beforehand.

The following list contains some examples, organized by topical areas, of what computational analysts need to do when comparing the code results with formal validation benchmarks and documenting the process for their own uses. Our intent in this list is to stress certain elements and to add new elements that may not have been called out previously in this paper.

- *Code verification*. References should be provided to document the code verification activities that have been completed and the version of the code that was used for these activities.
- *Solution verification*. Detailed information should be provided about iterative error convergence. At least three mesh resolutions and three temporal discretizations should be computed so that Richardson's method can be used to estimate the spatial and temporal discretization error on each of the SRQs that are compared with the experimental data. In addition, the observed order of convergence should be documented, along with the theoretical order of convergence.
- *Computation of SRQs*. In almost all fields of engineering, it is traditional to compute deterministic values for SRQs. That is, it is assumed that no uncertainty exists in any of the input quantities, e.g., boundary conditions, initial conditions, material properties, etc., so that a single value is computed for each of the SRQs. These deterministic values are then compared with the experimentally measured SRQs. This is, of course, the minimum level of comparison that should be made between code results and experimental benchmark results. It is recommended, however, that nondeterministic results be computed for each SRQ based on the uncertainty quoted for each input quantity, as stated in the validation benchmark. This is usually referred to as UQ of SRQs as a function of uncertain input quantities. As discussed in Section 4.1.3, the uncertain input quantity could be characterized as an interval, an imprecise probability distribution, or a precise probability distribution. Propagation of these uncertain quantities through the computational simulation model will likely rely on methods like Monte Carlo sampling or Latin hypercube sampling (Fishman, 1995; Robert, 1999; Helton and Davis, 2003; Santner et al., 2003). Importantly, major increases in computational resources will be required to compute tens or hundreds of solutions needed for the sampling techniques. In our experience, there will be much resistance to expending this level of computational resources for this purpose. Nonetheless, the probabilistic risk assessment communities, especially those concerned with nuclear reactor safety and the underground storage of nuclear waste, have accepted this philosophy of simulation for over two decades.
- *Validation metrics*. Traditional graphical comparisons should be included; however, validation metrics should also be used. Because validation metrics are in an early stage of development, only a limited range of examples are available (Dowding, 2001; Trucano et al., 2001, 2006; Hills and Trucano, 2002; Oberkampf and Trucano, 2002; Paez and Urbina, 2002; Hills and Leslie, 2003; Rutherford and Dowding, 2003; Chen et al., 2004; Dowding et al., 2004; Oberkampf and Barone, 2004, 2006; Mahadevan and Ramesh, 2005; Hills, 2006; Rebba et al., 2006). Validation metric results should be computed for *all* the SRQs measured in the experiment so that objective information is complete rather than partial or biased toward those that "look good."
- *Calibration*. Throughout this paper, we have carefully distinguished between *validation*, i.e., assessment of model accuracy, and *calibration*, i.e., activities to optimize model parameters when code results are compared with experimental measurements. Without a doubt, the most common parameters that are optimized are those that were not provided by the experimentalist in the documentation of the experiment. That is why we have stressed the importance of the *experimentalist* providing uncertainty estimates of *all* input quantities that might be needed for simulations. However, we recognize that there will probably be some "wiggle room" for computational analysts to optimize unmeasured, and undocumented, input quantities needed for the code that are related to physical characteristics of the experiment. If this is done in obtaining the code results, we believe it is necessary for the computational analyst to document any procedures that are used to optimize the input quantities. Our recommendation also applies to any numerical parameters (e.g., numerical damping, numerical smoothing, and hour-glass control of the vibrational modes of individual elements in solid dynamics meshes).
- *Global sensitivity analysis*. Here we mean an analysis that rank-orders the importance of each uncertain input for each SRQ according to the magnitude of change of the SRQ for a unit change in each uncertain input. This analysis is typically conducted by using the sampling results from the UQ analysis discussed above and reprocessing these results (see, for example, Saltelli et al., 2000, 2004; Ferson and Tucker, 2006; Helton et al., 2006). Conducting a sensitivity analysis as part of a comparison of code results with a validation benchmark is important from two perspectives. First, the analyst computing the results, or another analyst reading the documentation, will obtain a deeper understanding of the importance of different input quantities in relationship to the SRQs. Often, the

ranking of sensitivities can be quite surprising. Second, the experimentalist who conducted the experiment can use the sensitivity analysis to possibly update the uncertainty estimation on some measured quantities. Also, the experimentalist, or possibly a different experimental group, may choose to conduct a new experiment and judiciously reduce the experimental uncertainty on the largest contributors to uncertainty in the SRQs.

## 5. Implementation issues of a verification and validation database

If V&V strong-sense benchmarks (SSBs) and a database to house them were to become a reality, a number of complex and difficult implementation and organizational issues would need to be addressed. Some of these issues would be, for example, primary and secondary goals of the database, initial construction of the database, review and approval procedures for entries into the database, open versus restricted use of the database, the structure of the software framework for searching and retrieving information on SSBs in the database, organizational control of the database, relationship of the controlling organization to existing private and governmental organizations and engineering societies, and initial and long-term funding of the database. These issues are of major importance to the joint community of individuals, corporations, commercial software companies, nonprofit organizations, engineering societies, universities, and governmental organizations with serious interest in improving CS&E.

Initial construction of the database would be technically and organizationally complex, as well as costly. Populating the database with relevant, high-quality benchmarks would require a wide-ranging effort that cuts across major communities of applied mathematics, model building, experiment, computation, engineering applications, and business decision making. Putting this kind of collaborative effort together hinges on a careful plan that takes the long-term view for the database. The benchmark effort we describe in this paper is not feasible as a short-term task. Much of what we recommend clearly aims at a sustainable and long-term use of the database, with an implication that the quality and breadth of the database improves over a long period of time. The long-term success of the database requires a sound starting point, with broad consensus from all interested parties about goals, use, access, and funding over the long term.

Broad organizational issues must be addressed very early in the planning stage. For example, will a single organization (nonprofit, academic, or governmental) have responsibility for database maintenance, configuration management, and day-to-day operation? Will the database have a role beyond its immediate community, as we have essentially argued in this paper? *Broad impact* then implies that there is the goal of open access to the database for the good of the CS&E community, specifically the world community in each of the traditional scientific and engineering disciplines. But how is this goal compatible with the significant expense needed to create, maintain, and improve the database? Financial supporters and users of the database would need to be convinced of the value returned to them for their investment. The returned value could be in many forms, such as improvements in their software products, the ability to attract new customers to their software products, and use of the database as a quality assessment tool for organizations or government agencies to allow contractors to bid on new projects. If proprietary information is used in the database, we believe it would greatly diminish, possibly eliminate, the ability to create and sustain the database. Some have argued that the database could be constructed so that proprietary information could be segregated from generally available information. We believe that private corporations would not be convinced such segregation could be accomplished with high confidence.

It seems that V&V databases of the type we have discussed should be constructed along the lines of traditional engineering and science disciplines, e.g., fluid dynamics, solid dynamics, electrodynamics, neutron transport, plasma dynamics, and molecular dynamics. How each of these disciplines might begin to construct databases certainly depends on the traditions, applications, and funding sources in each of these fields. The nuclear power industry, for example, has a deeply embedded, long-term tradition of international cooperation. On the other hand, the aerospace industry, both aircraft and spacecraft builders, has a fierce competitive nature. We envision that a different implementation and database structure would be chosen in these two communities.

This paper focused on the construction of SSBs primarily for the purpose of assessing numerical accuracy in codes (verification) and of assessing physics modeling accuracy in codes (validation). We recognize this is a narrow view of the possible uses of benchmarks, but we believe that SSBs are critically needed at this early stage of maturity of computational simulation. We suggest that a secondary purpose for the establishment and use of SSBs is for the development of best practices in computational simulation. As recognized by NAFEMS (NAFEMS, 2006) and ERCOFTAC (Casey and Wintergerste, 2000), there is a compelling need for improvements in professional practice in computational simulation. In our opinion, a convincing argument could be made that the most common failures in industrial applications of computational simulation result from mistakes made by practitioners using the code. Corporate and governmental management, of course, shoulders the ultimate responsibility for mentoring and training these practitioners, as well as for monitoring their computational simulation work-products. Given the qualities of SSBs discussed previously, these benchmarks could be viewed as very carefully documented step-by-step sample problems from which practitioners, new and experienced, could learn a great deal.

Rizzi and Vos (1998) and Vos et al. (2002) discuss how validation databases could be built and used by a wide range of individuals and organizations. They stress the importance of close collaboration between corporations and universities in the construction and refinement of a validation database. In this regard, they also stress the value of workshops that are focused on specialty topics to improve the modeling efforts and simulations that are compared to experimental data. They discuss a number of workshops and initiatives in Europe, primarily funded by the European Union. Often, these workshops provide dra-

matic evidence of the power of carefully defined and applied V&V benchmarks. One such effort organized in the United States, but with participants from around the world, is the series of Drag Prediction Workshops (Levy et al., 2003; Hemsch, 2004; Hemsch and Morrison, 2004; Laflin et al., 2005; Rumsey et al., 2005). These workshops have been extraordinarily enlightening from two perspectives: (a) there was great variability in the drag predictions from computational analysts for a relatively simple aircraft geometry, and (b) there were surprisingly large differences between the computational results and the experimental measurements. The key factor in this exercise that resulted in a "surprisingly large range of results" is that this was a blind comparison. It was no surprise to us. Results from these types of workshops could form the basis for initial submittals of new V&V benchmarks into the database.

We believe an Internet-based system would provide the best vehicle for the deployment of V&V databases for three reasons. First, the ability to build, quickly share, and collaborate with an Internet-based system is now blatantly obvious. A paper-based system would be completely unworkable, as well as decades behind the current state of information technology. We speculate on one aspect of deployment, although this issue is beyond the purposes of this paper. Many businesses around the world are gaining a better understanding of the competitive advantage provided by the speed of information transfer within their organizations, even if their organizations are spread around the world. Thus, we expect that corporate acceptance of a benchmark effort would hinge on Internet deployment.

Second, descriptive terms that are of interest in a particular application of interest could be input to a search engine that could find all of the benchmarks that would contain those terms. The search engine could operate much like that found in Google or Wikipedia. Functionality could be expanded to include a relevancy-ranking feature that would further improve the search-and-retrieval capability. The overall system design would include configuration-, document-, and content-management elements. Then the benchmarks that were retrieved could be sorted according to their relevance to the words input to the search. One could then select the hyperlinks embedded within any of the benchmarks found. When a particular benchmark is displayed, it could have links from important words in the benchmark description to more detailed information in the benchmark.

And third, the computer-based system could instantly provide much more detail about each benchmark. We recommend that the documentation of V&V benchmarks be produced in an electronic format that is widely usable and robust across many computer operating systems. Of the electronic formats available, Adobe Portable Document Format (PDF) is the most commonly used and has many desirable characteristics; however, we also recommend that this format be supplemented with additional file formats for specialized information. For example, tabular data could be stored in ASCI text files or in Microsoft Excel files; high-resolution digital photographs should be stored in easily usable formats such as, tiff, PDF, and JPEG; digital video files should be stored in formats such as QuickTime, MPEG, or AVI; and computer software should be written in common languages such as C++, Fortran, or Java. The computer software would be necessary for documenting the source terms in the MMS.

In the long term, new validation experiments should be funded either by the organization controlling the database or by for-profit private, nonprofit, university, or governmental organizations. The organization controlling the database could receive its funding from subscribing members to the organization, and possibly from governmental funding. The funding could be directed to both operation and maintenance of the database and to constructing new V&V benchmarks. When new validation results are entered into the database, there would be a unique opportunity for blind comparisons. As we have stressed several times, blind comparisons are the real test of predictive-capability prowess. We believe that identification of new validation experiments should be the responsibility of both the application community and the database organization. The organizational role and facilitation of discussions regarding which experiments should be conducted is best served by the database organization. For example, the database organization could serve as an unbiased referee between for-profit corporations desiring more application-relevant experiments and model builders who are more knowledgeable of the weaknesses of modeling for complex systems.

## 6. Concluding remarks

In this paper, we have made the argument that significant improvements in the methodology and practice of V&V are necessary to achieve improved credibility in CS&E. We have discussed in detail one element of needed improvements: the design, construction, and use of strong-sense benchmarks (SSBs) in V&V. If the reader is of the opinion that CS&E is mature, fully capable, and reliable to shoulder the new responsibilities demanded of it, then you will have little interest in the ideas proposed here. Or, if you are of the opinion that V&V is "Too hard. Too slow. Too expensive," then you also will have little interest in our recommendations. If the reader is of the opinion, as are we, that CS&E is in its early stages of development and that its contributions to business, society, and governmental policies must be critically assessed and broadly available, then you will be more interested in our ideas. Even though the development of SSBs will be difficult, slow, and costly, they are necessary for the maturation of CS&E. This maturation is particularly important for the certification and regulatory oversight of the performance, safety, and reliability of high-consequence systems. In the past, the emphasis in certification and regulatory oversight, for example, in nuclear power reactors and large civil engineering projects, has been directed toward the physics modeling aspects of an analysis. We contend that the tools of CS&E, i.e., the computer codes, are becoming so complex that they too must be critically assessed. If SSBs can be developed and their pedigree documented to the level of being considered internationally as a "standard," then they can be used to make significant contributions to code assessment.

While we only touched on organizational issues surrounding the construction and use of V&V databases, these are, in fact, highly sensitive challenges and are rooted in different aspects of

worldwide economic competition, organizational and national prestige, and national security. Increasing the level of formality of V&V by constructing databases is going to inevitably lead to active discussions about further improvements in university education and professional-level training in the field of computational science. This is the inevitable consequence of devoting large amounts of expert thought, money, and labor to the deployment and utilization of such databases. If these databases are developed and widely used around the world, then they are going to evolve into *de facto*, if not intentionally accepted, standards. There would be similarities of V&V benchmark standards to those international procedures that have developed over the last century for physical measurement standards. However, the range of expert knowledge required for V&V benchmark standards would be much broader than the knowledge requirements for physical measurement standards.

## Acknowledgements

The authors thank Sam Key, Curtis Ober, and Patrick Knupp, all of Sandia National Laboratories, for reading a draft of this paper and providing a number of constructive suggestions for improvements. We thank Patrick Roache, a private consultant, for providing detailed comments and suggestions for improving the manuscript. We also thank Rhonda Reinert of Technically Write for providing extensive editorial assistance during the writing of this 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-AC04-94AL85000.

## References

Abanto, J., Pelletier, D., Garon, A., Trepanier, J.-Y., Reggio, M., 2005. Verification of Some Commercial CFD Codes on Atypical CFD Problems. 43rd AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, AIAA Paper 2005-0682.

ABAQUS, 2006. ABAQUS Benchmarks Manual. ABAQUS Inc.

Aeschliman, D.P., Oberkampf, W.L., 1998. Experimental methodology for computational fluid dynamics code validation. AIAA J. 36 (5), 733–741.

AIAA, 1998. Guide for the Verification and Validation of Computational Fluid Dynamics Simulations. American Institute of Aeronautics and Astronautics, AIAA-G-077-1998.

Ainsworth, M., Oden, J.T., 2000. A Posteriori Error Estimation in Finite Element Analysis. John Wiley, New York.

ANS, 1987. Guidelines for the Verification and Validation of Scientific and Engineering Computer Programs for the Nuclear Industry. American Nuclear Society, ANSI/ANS-10.4-1987.

ANSYS, 2005. ANSYS Verification Manual. ANSYS, Inc, Release 10.0.

ASME, 2006. Guide for Verification and Validation in Computational Solid Mechanics. American Society of Mechanical Engineers, ASME V&V 10-2006.

Axelsson, O., 1996. Iterative Solution Methods. Cambridge University Press, Cambridge, UK.

Babuska, I., Strouboulis, T., 2001. The Finite Element Method and its Reliability. Oxford University Press, Oxford, UK.

Beven, K., 2002. Towards a coherent philosophy of modelling the environment. Proc. R. Soc. Lond. Ser. A 458 (2026), 2465–2484.

Blottner, F.G., 1982. Influence of boundary approximations and conditions on finite-difference solutions. J. Comput. Phys. 48 (2), 246–269.

Botella, O., 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.

Box, G.E.P., Hunter, J.S., Hunter, W.G., 2005. Statistics for Experimenters: Design, Innovation, and Discovery. John Wiley, New York.

Carpenter, M.H., Casper, J.H., 1999. Accuracy of Shock Capturing in Two Spatial Dimensions. AIAA J. 37 (9), 1072–1079.

Casey, M., 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.

Chen, W., Baghdasaryan, L., Buranathiti, T., Cao, J., 2004. Model validation via uncertainty propagation. AIAA J. 42 (7), 1406–1415.

Coleman, H.W., Steele Jr., W.G., 1999. Experimentation and Uncertainty Analysis for Engineers. John Wiley, New York.

Diskin, B., Thomas, J.L., 2002. Analysis of Boundary Conditions for Factorizable Discretizations of the Euler Equations. NASA/ICASE, NASA/CR-2002-211648.

DoD, 1996. DoD Instruction 5000.61: Modeling and Simulation (M&S) Verification, Validation, and Accreditation (VV&A). From www.dmso.mil/docslib.

DoD, 1996. Verification, Validation, and Accreditation (VV&A) Recommended Practices Guide. From www.dmso.mil/docslib.

Dowding, K., 2001. Quantitative Validation of Mathematical Models. ASME International Mechanical Engineering Congress Exposition. American Society of Mechanical Engineers, New York.

Dowding, K.J., Hills, R.G., Leslie, I., Pilch, M., Rutherford, B.M., et al., 2004. Case Study for Model Validation: Assessing a Model for Thermal Decomposition of Polyurethane Foam. Sandia National Laboratories, SAND2004-3632.

Eca, L., Hoekstra, M., 2006a. An Introduction to CFD Code Verification Including Eddy-Viscosity Models. ECCOMAS CFD 2006, Egmond aan Zee. European Community on Computational Methods in Applied Sciences, The Netherlands.

Eca, L., Hoekstra, M., 2006b. Verification of Turbulence Models with a Manufactured Solution. ECCOMAS CFD 2006, Egmond aan Zee. European Community on Computational Methods in Applied Sciences, The Netherlands.

Eca, L., Hoekstra, M., Hay, A., Pelletier, D., Roache, P.J., 2005. A Manufactured Solution for a Two-Dimensional Steady Wall-Bounded Incompressible Turbulent Flow. Instituto Superior Tecnico, IST Report D72-34.

ERCOFTAC, 2000. Portal to Fluid Dynamics Database Resources. From http://ercoftac.mech.surrey.ac.uk.

Ferson, S., Tucker, W.T., 2006. Sensitivity in Risk Analyses with Uncertain Numbers. Sandia National Laboratories, SAND2006-2801.

Ferziger, J.H., Peric, M., 1996. Computational Methods for Fluid Dynamics. Springer-Verlag, New York.

Fishman, G.S., 1995. Monte Carlo: Concepts, Algorithms, and Applications. Springer, New York.

Helton, J.C., Davis, F.J., 2003. Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems. Reliab. Eng. Syst. Saf. 81 (1), 23–69.

Helton, J.C., Johnson, J.D., Sallaberry, C.J., Storlie, C.B., 2006. Survey of sampling-based methods for uncertainty and sensitivity analysis. Reliab. Eng. Syst. Saf. 91 (10–11), 1175–1209.

Hemsch, M., 2004. Statistical analysis of computational fluid dynamic solutions from the Drag Prediction Workshop. J. Aircraft 41 (1), 95–103.

Hemsch, M., Morrison, J.H., 2004. Statistical Analysis of CFD Solutions from 2nd Drag Prediction Workshop. 42nd AIAA Aerospace Sciences Meeting and Exhibit. American Institute of Aeronautics and Astronautics, Reno, NV.

Hills, R.G., 2006. Model validation: model parameter and measurement uncertainty. J. Heat Transfer 128 (4), 339–351.

Hills, R.G., Leslie, I., 2003. Statistical Validation of Engineering and Scientific Models: Validation Experiments to Application. Sandia National Laboratories, SAND2003-0706.

Hills, R.G., Trucano, T.G., 2002. Statistical Validation of Engineering and Scientific Models: A Maximum Likelihood Based Metric. Sandia National Laboratories, SAND2001-1783.

Hirsch, C., 1988. Numerical Computation of Internal and External Flows Fundamentals of Numerical Discretization, vol. 1. John Wiley, New York.

Hirsch, C., 1990. Numerical Computation of Internal and External Flows: Computational Methods for Inviscid and Viscous Flows, vol. 2. John Wiley, New York.

IEEE, 1984. IEEE Standard Dictionary of Electrical and Electronics Terms, ANSI/IEEE Std 100-1984.

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. International Standards Organization.

Keller, H.B., 1969. Accurate difference methods for linear ordinary differential systems subject to linear constraints. SIAM J. Numer. Anal. 6, 8–30.

Knupp, P., Salari, K., 2002. Verification of Computer Codes in Computational Science and Engineering. Chapman & Hall/CRC, Boca Raton, FL.

Kusnezov, D.F., 2004. Advanced Simulation & Computing: The Next Ten Years. Sandia National Laboratories, SAND2004-3740P.

Laflin, K.R., Klausmeyer, S.M., Zickuhr, T., Vassberg, J.C., Wahls, R.A., et al., 2005. Data summary from the second AIAA computational fluid dynamics drag prediction workshop. J. Aircraft 42 (5), 1165–1178.

Laney, C.B., 1998. Computational Gasdynamics. Cambridge University Press, Cambridge, UK.

Levy, D.W., Zickuhr, T., Wahls, R.A., Pirzadeh, S., Hemsch, M.J., 2003. Data summary from the first AIAA computational fluid dynamics drag prediction workshop. J. Aircraft 40 (5), 875–882.

Mahadevan, S., Ramesh, R., 2005. Validation of reliability computational models using Bayes networks. Reliab. Eng. Syst. Saf. 87, 223–232.

Mehta, U.B., 1995. Guide to Credible Computational Fluid Dynamics Simulations. 26th AIAA Fluid Dynamics Conference. American Institute of Aeronautics and Astronautics, San Diego, CA, AIAA Paper 95-2225.

Montgomery, D.C., 2000. Design and Analysis of Experiments. John Wiley, Hoboken, NJ.

Morton, K.W., 1996. Numerical Solution of Convection-Diffusion Problems. CRC Press, Boca Raton, FL.

NAFEMS, 2006. NAFEMS Website. From www.NAFEMS.org.

NEA, 1977. Loss of Coolant Accident Standard Problems. Nuclear Energy Agency, Committee on the Safety of Nuclear Installations, Report No. 17.

NEA, 2004. CSNI International Standard Problem Procedures. Nuclear Energy Agency, Committee on the Safety of Nuclear Installations, Report No. 17-revision 4.

NPARC, 2000. CFD Verification &Validation: NPARC Alliance. From http://www.grc.nasa.gov/WWW/wind/valid/homepage.html.

Oberkampf, W.L., 1994. A Proposed Framework for Computational Fluid Dynamics Code Calibration/Validation. 18th AIAA Aerospace Ground Testing Conference. American Institute of Aeronautics and Astronautics, Colorado Springs, CO, AIAA Paper 94-2540.

Oberkampf, W.L., Aeschliman, D.P., 1992. Joint computational/experimental aerodynamics research on a hypersonic vehicle: Part 1. Experimental results. AIAA J. 30 (8), 2000–2009.

Oberkampf, W.L., Aeschliman, D.P., Henfling, J.F., Larson, D.E., 1995. Surface Pressure Measurements for CFD Code Validation in Hypersonic Flow. 26th AIAA Fluid Dynamics Conference. American Institute of Aeronautics and Astronautics, San Diego, CA, AIAA Paper 95-2273.

Oberkampf, W.L., Aeschliman, D.P., Tate, R.E., Henfling, J.F., 1993. Experimental Aerodynamics Research on a Hypersonic Vehicle. Sandia National Laboratories, SAND92-1411.

Oberkampf, W.L., Barone, M.F., 2004. Measures of Agreement Between Computation and Experiment: Validation Metrics. 34th AIAA Fluid Dynamics Conference. American Institute of Aeronautics and Astronautics, Portland, OR, AIAA Paper 2004-2626.

Oberkampf, W.L., Barone, M.F., 2006. Measures of agreement between computation and experiment: validation metrics. J. Comput. Phys. 217 (1), 5–36.

Oberkampf, W.L., Blottner, F.G., 1998. Issues in computational fluid dynamics code verification and validation. AIAA J. 36 (5), 687–695.

Oberkampf, W.L., Trucano, T.G., 2002. Verification and validation in computational fluid dynamics. Prog. Aerospace Sci. 38 (3), 209–272.

Oberkampf, W.L., Trucano, T.G., Hirsch, C., 2004. Verification, validation, and predictive capability in computational engineering and physics. Appl. Mech. Rev. 57 (5), 345–384.

Oden, J.T., 1993. In: Whiteman, J.R. (Ed.), Error Estimation and Control in Computational Fluid Dynamics. The Mathematics of Finite Elements and Applications. John Wiley, New York, pp. 1–23.

Pace, D.K., 1998. Synopsis of Fidelity Ideas and Issues. 1998 Spring Simulation Interoperatibility Workshop Papers.

Paez, T.L., Urbina, A., 2002. Validation of Mathematical Models of Complex Structural Dynamic Systems. Proceedings of the Ninth International Congress on Sound and Vibration. International Institute of Acoustics and Vibration, Orlando, FL.

Prabhakar, V., Reddy, J.N., 2006. Spectral/hp penalty least-squares finite element formulation for the steady incompressible Navier-Stokes Equations. J. Comput. Phys. 215 (1), 274–297.

QNET-CFD, 2001. Thematic Network on Quality and Trust for the Industrial Applications of CFD. From www.qnet-cfd.net.

Rebba, R., Mahadevan, S., Huang, S., 2006. Validation and error estimation of computational models. Reliab. Eng. Syst. Saf. 91 (10–11), 1390–1397.

Refsgaard, J.C., Henriksen, H.J., 2004. Modelling guidelines-terminology and guiding principles. Adv. Water Resour. 27, 71–82.

Rizzi, A., Vos, J., 1998. Toward establishing credibility in computational fluid dynamics simulations. AIAA J. 36 (5), 668–675.

Roache, P.J., 1990. Need for control of numerical accuracy. J. Spacecraft Rockets 27 (2), 98–102.

Roache, P.J., 1994. Perspective: a method for uniform reporting of grid refinement studies. J. Fluids Eng. 116, 405–413.

Roache, P.J., 1997. Quantification of Uncertainty in Computational Fluid Dynamics. Annual Review of Fluid Mechanics. J.L. Lumley, M. Van Dyke, Palo Alto, CA. Annu. Rev. 29, 126–160.

Roache, P.J., 1998a. Verification and Validation in Computational Science and Engineering. Hermosa Publishers, Albuquerque, NM.

Roache, P.J., 1998b. Verification of codes and calculations. AIAA J. 36 (5), 696–702.

Robert, C.P., 1999. Monte Carlo Statistical Methods. Springer-Verlag, New York.

Roy, C.J., 2004. Verification of Euler/Navier-Stokes codes using the method of manufactured solutions. Int. J. Numer. Methods Fluids 44 (6), 599–620.

Roy, C.J., Blottner, F.B., 2001. Assessment of one- and two-equation turbulence models for hypersonic flows. J. Spacecraft Rockets 38 (5), 699–710.

Roy, C.J., McWherter-Payne, M.A., Oberkampf, W.L., 2000. Verification and Validation for Laminar Hypersonic Flowfields. Fluids 2000 Conference. American Institute of Aeronautics and Astronautics, Denver, CO, AIAA Paper 2000-2550.

Roy, C.J., McWherter-Payne, M.A., Oberkampf, W.L., 2003a. Verification and validation for laminar hypersonic flowfields. Part 1: Verification. AIAA J. 41 (10), 1934–1943.

Roy, C.J., Oberkampf, W.L., McWherter-Payne, M.A., 2003b. Verification and validation for laminar hypersonic flowfields. Part 2: Validation. AIAA J. 41 (10), 1944–1954.

Rumsey, C.L., Rivers, S.M., Morrison, J.H., 2005. Study of CFD variation on transport configurations for the second drag-prediction workshop. Comput. Fluids 34 (7), 785–816.

Rutherford, B.M., Dowding, K.J., 2003. An Approach to Model Validation and Model-Based Prediction—Polyurethane Foam Case Study. Sandia National Laboratories, SAND2003-2336.

Rykiel, E.J., 1996. Testing Ecological Models: the meaning of validation. Ecol. Modell. 90 (3), 229–244.

Saltelli, A., Chan, K., Scott, E.M. (Eds.), 2000. Sensitivity Analysis. Wiley, New York.

Saltelli, A., Tarantola, S., Campolongo, F., Ratto, M., 2004. Sensitivity Analysis in Practice: A Guide to Assessing Scientific Models. John Wiley, Chichester, England.

Santner, T.J., Williams, B.J., Notz, W., 2003. The Design and Analysis of Computer Experiments. Springer, New York.

Soudah, J., Doebling, S., Sefcik, J., Pilch, M., Trucano, T. ASC V&V Program Strategy: Toward a Predictive Enterprise. National Nuclear Security Administration, in preparation.

Srivastava, B.N., Werle, M.J., Davis, R.T., 1979. A finite difference technique involving discontinuous derivatives. Comput. Fluids 7 (1), 69–74.

Trucano, T.G., Easterling, R.G., Dowding, K.J., Paez, T.L., Urbina, A., et al., 2001. Description of the Sandia Validation Metrics Project. Sandia National Laboratories, SAND2001-1339.

Trucano, T.G., Pilch, M., Oberkampf, W.L., 2002. General Concepts for Experimental Validation of ASCI Code Applications. Sandia National Laboratories, SAND2002-0341.

Trucano, T.G., Pilch, M., Oberkampf, W.L., 2003. On the Role of Code Comparisons in Verification and Validation. Sandia National Laboratories, SAND2003-2752.

Trucano, T.G., Post, D.E., Pilch, M., Oberkampf, W.L., 2005. Software Engineering Intersection with Verification and Validation of Higher Performance Computational Science Software: Some Observations. Sandia National Laboratories, SAND2005-3662P.

Trucano, T.G., Swiler, L.P., Igusa, T., Oberkampf, W.L., Pilch, M., 2006. Calibration, validation, and sensitivity analysis: What's What. Reliab. Eng. Syst. Saf. 91 (10–11), 1331–1357.

Turkel, E., 1986. Accuracy of schemes with nonuniform meshes for compressible fluid-flows. Appl. Numer. Math. 2 (6), 529–550.

Vos, J.B., Rizzi, A., Darracq, D., Hirschel, E.H., 2002. Navier-Stokes solvers in european aircraft design. Prog. Aerospace Sci. 38 (8), 601–697.

Walker, M.A., Oberkampf, W.L., 1992. Joint computational/experimental aerodynamics research on a hypersonic vehicle: Part 2. Computational results. AIAA J. 30 (8), 2010–2016.

White, F.M., 1991. Viscous Fluid Flow. McGraw Hill, New York.