AIAA JOURNAL
Vol. 36, No. 5, May 1998

Verification of Codes and Calculations

Patrick J. Roache*
Albuquerque, New Mexico 87031

Background discussion, definitions, and descriptions are given for some terms related to confidence building in computational fluid dynamics. The two principal distinctions made are between verification vs validation and between verification of codes vs verification of individual calculations. Also discussed are numerical errors vs conceptual modeling errors; iterative convergence vs grid convergence (or residual accuracy vs discretization accuracy); confirmation, calibration, tuning, and certification; error taxonomies; and customer illusions vs customer care. Emphasis is given to rigorous code verification via systematic grid convergence using the method of manufactured solutions, and a simple method for uniform reporting of grid convergence studies using the Grid Convergence Index (GCI). Also discussed are surrogate single-grid error indicators.

Introduction

VERIFICATION and the broader area of confidence building in computational fluid dynamics (CFD) is a curious subject. The growing recognition of its importance is attested to by the policy statements given by the American Society of Mechanical Engineers[1–3] and AIAA[4] and in other journals.[5] (See Ref. 6 for my experience in implementing the policy in Ref. 1.) The tone of articles can be fairly legalistic, yet the area is quite subjective, dependent on opinions, world view, philosophy of science, philosophy of engineering, and appeals based on common sense as much as mathematics. This is partly due to the semantics involved.

Semantics

Semantic distinctions, in my experience, can have major project consequences. For example, the choice of "verification" or "validation" was originally arbitrary and is now recommended solely because of common developing use. In a common English thesaurus, "verify," "validate," and "confirm" are all synonyms, but the words are used herein, and generally in code quality assurance (QA), as technical terms with more context-specific meaning.

This is not a universally accepted attitude toward semantics. In a widely quoted[7,8] paper, Oreskes et al.[9] think that we can find the real meaning of a technical term by inquiring about its common meaning. They make much of supposed intrinsic meaning in verify and validate and agonize over truth. They come to the remarkable conclusion that it is impossible to verify or to validate a numerical model of a natural system! They clearly have no intuitive concept of error tolerance, or of range of applicability, or of common sense. Clearly, we are interested in practical definitions, applied in the context of engineering and science accuracy, not in such worthless semantics and effete philosophizing.

In the following, definitions and descriptions are given for some terms related to confidence building in CFD. (I will not try to define confidence building with any precision.)

Code Verification and Validation: Numerical vs Conceptual Modeling

First and foremost, we must repeat the essential distinction between code verification and validation. Following Boehm[10] and Blottner,[11] we adopt the succinct description of verification as "solving the equations right" and of validation as "solving the right equations." The code author defines precisely what partial differential equations are being solved and convincingly demonstrates that they are solved correctly, i.e., usually with some order of accuracy, and always consistently, so that as some measure of discretization Δ (e.g., the mesh increments) $\rightarrow 0$ the code produces a solution to the continuum equations; this is verification. Whether or not those equations and that solution bear any relation to a physical problem of interest to the code user is the subject of validation.

In a meaningful sense, a code cannot be validated, but only a calculation (or range of calculations with a code) can be validated. In my experience, dealing with other than algorithm developers, this is a difficult concept and requires continual reiteration. Validation can be a multistep process and must include, for complex physical problems, a list of features of the governing equations that have been validated.

Another way to make the distinction, i.e., to get to the idea behind the words, beyond mere semantics, is to speak of numerical errors vs conceptual modeling errors. An example of conceptual modeling vs numerical modeling is the assumption of incompressibility. This is clearly a conceptual modeling assumption. Is it the code builder's fault or any criticism of the code itself if the user incorrectly applies it? For example, dynamic stall involves compressibility at a surprisingly low freestream Mach number. Results from an incompressible code may not agree with experiment very well, but we cannot say that the code fails verification because it was applied to compressible flow, although we may have some sympathy for the user who is fooled by dynamic stall. But no one would have sympathy for a user who applied an incompressible flow code to a re-entry vehicle at Mach 20. The lack of agreement with experiment is not a code problem but a modeling problem. The same is true of many practical aspects of applying a CFD code. The model includes more than the code. It includes conceptual modeling assumptions, e.g., incompressibility, symmetry, etc. It also includes data input to the code, e.g., geometry data, which are not so easy to determine accurately as many people assume, and boundary conditions and initial conditions. These can lead to failure of validation of a model with possibly no criticism of the code.

The typical computer science view of code verification is not mine. Verification in my view does not include all aspects of code QA or confidence building. For example, verification does not include the important and nagging concerns of version control, or archiving of input data, or documentation (external and internal). Likewise, extensive code application in a user community builds confidence but is not part of verification per se. Less obviously, in my opinion, verification defined as a technical term herein would not include reading of source code. Blottner[11] described his verification of a Navier–Stokes code and included the fact that the Fortran source code was examined. Although perhaps that is useful and contributes to confidence building, I do not consider it to be part of code verification per se (nor of confirmation; see the following text). Reading the code could usefully ascertain (or verify in the general, nontechnical sense) that each subprogram is performing its intended task and therefore could be required as part of a QA certification. However, we can read source code, either our own or someone else's,

Presented as Paper 95-2224 at the AIAA 26th Fluid Dynamics Conference, San Diego, CA, June 19–22, 1995; received Oct. 22, 1996; revision received Feb. 6, 1998; accepted for publication Feb. 6, 1998.

*Consultant, P.O. Box 9229. Senior Member AIAA.

interminably and still not be able to claim verification. On the other hand, if the grid convergence tests prove that the code is, for example, second-order accurate, it would take some strange coding to nullify this test. (Perhaps some in-line limit on the parameter range that was not hit in the grid convergence test is possible.) Except for these unusual cases, reading of the code appears to me to be neither necessary nor sufficient for verification.

Surely few would claim that reading of code is any substitute for verification via grid convergence testing (although our customers have tried). If we included reading of the source code as part of code verification, then the code author's reading could be claimed as a partial verification. We know it does not prove a thing. In my view, verification (in this restricted, technical meaning) should treat the code as a black box.

Note I do not consider a code bomb, i.e., a divide by zero, or instability, to be a failure of code verification in the present context. As long as the code does not lie to the user, code verification has not been breached. Code robustness is certainly desirable and is properly part of QA and overall code evaluation, but in my use of the terms it is not an issue in verification. (Also, it is well known that robustness is often achieved at the expense of numerical accuracy.)

Also, I believe strongly that code verification can and should be completed (it is not an ongoing exercise) without physical experiments. Experimental agreement is part of validation, not verification, and the concepts are distinct enough to be worth preserving. Verification and validation are as distinct as mathematics and physics, on which they are based.

Validation has highest priority (to engineers and scientists) because nature is the final jury. But any experience with laboratory experiments will quickly disavow the absolute quality of experimental data. (It is asking too much of a CFD code to agree with wind-tunnel test data when these data do not agree with other wind-tunnel data, nor even from one run to another in the same wind tunnel.) Thus, I strongly believe that complete verification of a code should precede any comparisons with experimental data. (However, failure of a code to agree with an experimental benchmark with well-established error bounds would obviously raise suspicions about either the code or the experiments.)

This brings up another complaint with experiments. One sometimes hears complaints that a "CFD code needs additional data" that experimentors typically do not measure. For example, data are published on turbulent flow in channel expansions that do not characterize the details of the boundary layer ahead of the expansion (nor sometimes elementary quantities like boundary-layer thickness). But a CFD code needs no more information than the physics (and usually less, due to simplifying approximations in the turbulence theory). If experimenters have not measured these quantities, then they have an uncontrolled experiment, regardless of whether CFD codes will be used. The question, of course, is whether these unmeasured quantities are important to the physics of interest. This seems to me to be a major opportunity for CFD to contribute to experimental work; CFD can be used to predict the importance of difficult-to-measure quantities like wind-tunnel flow angularity, nonuniform stagnation enthalpy, surface waviness, etc., as discussed lucidly by Aeshliman et al.[12] A premier example given by Haynes et al.[13] is the sensitivity of boundary-layer transition to freestream vorticity, which is "an unusually difficult experiment" but which can be modeled by CFD.

Verification of Calculations

The second major distinction to be made is less recognized than the distinction between verification and validation. It is the distinction between verification of codes vs verification of (individual) calculations.

A code may be rigorously verified to be, for example, second-order accurate, but when the code is applied to a new problem, this fact provides no estimate of accuracy or confidence interval. It is still necessary to perform grid convergence tests to band the numerical error. (It would be preferable to have different words for these two verification activities, but I am at a loss for a clarifying term.) The very important point, independent of the semantics, is that use of a verified code is not enough. This point is probably well recognized by present readers but is not universally so. Especially in the commercial CFD arena, user expectations are often that purchase and use of a "really good code" will remove from the user the obligation of "doing his homework," i.e., the tedious work of verification of calculations via systematic grid convergence studies. This unrealistic expectation is sometimes encouraged by salespeople and by advertising.

Code Confirmation and Certification

Some computer science or QA people would have code verification necessarily performed by someone other than the code developer, sort of an arm's length transaction philosophy. In my view, it is ridiculous to expect code builders to not verify their code! Verification is a necessary part of code development. Code authors would be remiss in their duty if they released a code without verification. I would trust a code builder's verification results if presented in full; fraud is not usually the issue. But if it is, and if further tests are required (or repeats of the original verifications to check for fraud), then I would suggest the use of the term confirmation for calculations independently run by someone other than the code builder. Also, the suite of problems can be used for reverification (e.g., after porting to a new computer or a new compiler or to run through after the addition of new options) and for user training.

We[14] recognize five distinct regimes where errors can be made in benchmarking a CFD code, even without considering the validation question of whether the right equations are being solved for the target problem. The error regimes are 1) in code generation (either by hand or using computer symbolic manipulation; see the following text), 2) in code instructions, e.g., in a user manual or comment cards, 3) in problem setup, 4) in the definition and coding of a benchmark case (analytical solutions are often more difficult to code than numerical solutions), and 5) in the interpretation of code results. The first two are errors of the code author. The last three are errors of the code user, although ambiguous or scant code documentation can put some of the responsibility back onto the code author. Verification of a code removes regime 1 and, if done thoroughly, regime 2, but 3–5 still contain the potential for errors in any new application. We reluctantly conclude that there will be a continuing need for users to construct and exercise benchmark cases even when using verified codes.

I am not so clear on the concept of code certification, and I defer to others[15–20] for their definitions. My inclination would be to include in code certification the activity of code verification independent of the code authors (confirmation), other aspects of code QA including documentation, and validation for a project-oriented range of parameters.

Grid Convergence vs Iterative Convergence

The literature commonly uses the term convergence in two completely different ways. Readers of this paper will know the distinction between iterative convergence vs grid convergence (or residual accuracy vs discretization accuracy). Usually, the meaning is clear from the context, but sometimes confusion occurs, e.g., when some new variant of the SIMPLE algorithm[21] is presented as being more accurate. The accuracy claimed here is residual accuracy, i.e., what is better called iterative convergence accuracy or iterative speed, and has nothing to do with the order of accuracy of the discretization. Also, inadequate iterative convergence will pollute grid convergence results.

For the present subject, we note that iterative convergence can muddy the distinction between code verification and calculation verification because iterative tuning parameters, e.g., multigrid cycles, relaxation factors, etc., can be problem dependent.

Error Taxonomies

Several taxonomies of errors given in the literature are inadequate and misleading, in my opinion. Not all lists are taxonomies. For example, so-called "grid generation errors"[22] are not separate from discretization errors. For the verification of a code or a calculation, there are no such things as grid generation errors (nor are there "errors associated with coordinate transformations"[22]). Indeed, bad grids add to discretization error size but do not add new terms. This does not mean that one grid is as good as another, or that a really bad grid cannot magnify errors, but only that these so-called grid generation errors do not have to be considered separately from other

discretization errors in a grid convergence test. If the grid convergence test is performed, and the errors are shown to reduce as $\mathcal{O}(\Delta^2)$, for example, then all discretization errors are verified. One does not need to separately estimate or band the grid generation errors.

Likewise for the proposed numerical error bar of Refs. 23 and 24, which consists of separately estimated numerical errors from boundary conditions, computational domain size, temporal errors, and spatial errors. This is clearly a false taxonomy. Numerical errors at boundaries can be ordered in Δ (e.g., the various approaches for wall vorticity[25]) or possibly can be nonordered in Δ, e.g., the boundary-layer-like $\partial p/\partial n = 0$ or the downstream (outflow) boundary location. Ordered errors will tend to zero as the discretization improves, so that a boundary error from wall vorticity evaluation need not be considered separately from the grid convergence study. That taxonomy[23] already includes temporal errors and spatial errors and computational domain size errors, so that both ordered and nonordered boundary errors are already counted elsewhere in the taxonomy. Because the intention in Ref. 23 is to provide a quantitative breakdown in the sources of numerical error in an error band, the proposed taxonomy is not only confused but misleading.

Note, however, that outflow boundary errors may prove to be ordered not in Δ but in $1/l$ where l is the distance from the region of interest, e.g., an airfoil, to the outflow boundary. (See Zingg's data,[26] shown in Ref. 27 to be first order in $1/l$.)

The subject of outflow boundary conditions does produce some fuzziness in categorization of verification vs validation, in my opinion. The error can be ordered, as before, and therefore can be part of verification. That is, it is up to the user (who is doing the conceptual modeling) to estimate or band the error caused by the position of the outflow boundary. But if the code has some sophisticated outflow condition, e.g., a simple vortex condition for Euler equations, then the distinction is not so clear. Certainly the equations used are clear, and the code may solve the equations right (i.e., verification), yet there exists another benchmark solely from the mathematics (the case with infinite boundary distance) that could be used to justify the outflow condition without recourse to physical experiment (which would clearly be validation).

Another example of semantic failure or fuzzy taxonomy arises when we consider benchmarking a turbulent boundary-layer code or Parabolized Navier–Stokes (PNS) code against a Reynolds-stress averaged full Navier–Stokes (NS) code. Presume that both codes are convincingly "verified," i.e., they correctly solve their respective equations. Suppose that the PNS code results agree well with the NS code results for some range of parameters, e.g., including angle of attack. This agreement is not included in the term "verification" because the verification of the PNS code has already been completed prior to the NS benchmarking. Then we could say that the agreement has demonstrated that the PNS code is "solving the right equations" in one sense; i.e., it justifies the use of parabolic marching equations. Yet to claim "validation" would be overreaching because we have not demonstrated the adequacy of the turbulence model by comparison with experiment. We have "solved the right equations" in an intermediate sense of demonstrating that the PNS equations adequately represent the full NS equations but not in the ultimate sense of "solving the right *physical* equations." In situations dictating a legalistic distinction, one could claim "justification" of the simplification of mathematical models (full Navier–Stokes to boundary layer, compressible to incompressible, variable property to constant property, etc.) while making it clear that the physical "validation" remains to be accomplished.

Unfortunately, such "mere semantics" may become of vital interest when dealing with regulatory agencies, such as the Environmental Protection Agency, or with legal definitions in a NASA contract.

Truncation Error vs Discretization Error

Truncation error is an unfortunate term. Strictly speaking, it refers to the truncation at some finite value of a series, which could be analytical, e.g., the Sudicky–Frind solution,[28] or more commonly, in the present context, to the basis of developing the finite difference equations, the Taylor series expansion of the solution. It is a worthwhile concept because it allows one to define the order of the finite difference (or finite element, finite volume, etc.) method. Unfortunately, it is often used loosely in the sense of discretization error, i.e., the error that is caused by the fact that we can only use a finite number of grid points (or another measure of discretization). In a finite difference method (FDM), one cannot take the limit of infinite order (i.e., limit of zero truncation error) without also taking the limit of infinite number of grid points because high-order methods require higher support. This terminology makes the limit process somewhat misleading, in my opinion. Also, it confuses the issue of solution smoothness with discretization error because the Taylor series expansion depends on smoothness.

In the context of grid convergence tests, it is preferable to not speak of evaluating the truncation error of a numerical approximate solution but rather the discretization error that arises because of the finite discretization of the problem. This terminology applies to every consistent methodology, FDM, finite volume method (FVM), finite element method (FEM), spectral, pseudospectral, vortex-in-cell, etc., regardless of solution smoothness. (By consistent we mean, of course, that the continuum equations are recovered in the limit of infinite discretization.) The term truncation error is then reserved just for the examination of the order of convergence rate of the discretization. Note again the point of a taxonomy; these two errors are not independent. For any finite grid calculation, we do not have a truncation error (arising from the use of finite ordered FDM, for example) that we add to the discretization error (arising from the use of a finite number of grid points). And it is not possible to approach the limit of zero truncation error by arbitrarily increasing the order of the FDM or FEM without increasing the discretization. (Note that in the FEM we could fix a finite number of elements, but we would still have to increase the discretization, i.e., the support within the elements.) However, the alternate is true: we can in fact approach the limit of eliminating all the discretization error by arbitrarily increasing the number of grid points without changing the order of the method. Thus, discretization error is the preferable term for speaking of the numerical error of a calculation, and truncation error is not separate in the taxonomy of an error estimate for a calculation. (However, the order of the truncation error is still verified in the code verification.)

Truncation error has also been defined[22] as the residual resulting from substitution of the continuum (exact) solution values into the discrete equations on some grid. This definition can be useful for analysis of discrete methods. But, again, it is not distinguishable from discretization error in an accuracy estimate of a real calculation.

Calibration and Tuning

I prefer to use the term code calibration to mean the adjustment of parameters needed to fit experimental data, e.g., the six closure coefficient values necessary for two-equation turbulence models. However, colleagues assure me that the term is used in experimental studies just as a means of ascertaining accuracy, e.g., of a pressure probe. If extended to codes,[12] this definition, in my mind, makes code calibration almost indistinguishable from validation, or perhaps validation for a more restricted range of parameters.

The adjustment of parameters can be called tuning, which has a faintly pejorative association, deservedly if every new data set requires retuning, unfairly if reasonable universality is obtained.

Customer Illusions vs Customer Care

Catering to customer illusions can be antithetical to true customer care. This distinction is pertinent to CFD in code user training and education. The limits of applicability of CFD codes must be made clear. A CFD code cannot be an aerodynamicist on a chip; i.e., the code cannot substitute for user expertise in the technical area. True customer care is most important in the area of code robustness. We all know how to build a robust CFD code, following what has become a CFD joke.[29] "The good news about first-order upstream differencing is that it always gives you an answer. The bad news is that it always gives you the same answer, no matter what the Reynolds number." It is remarkable and disheartening to see how many industrial CFD practitioners will freely admit to caring little about numerical accuracy. For those who do, we must not gloss over the limitations of CFD and give them a false sense of security.

Other Distinctions

Another way to make distinctions is between the code author or code builder (which may of course be a team, including algorithm developers and programmers) and the code user or analyst. In many past situations, the same person performed both functions and therefore took the blame or credit for all aspects, but it is still worthwhile to make the distinction in functionality, especially with the rise of commercial general-purpose CFD software.

Another distinction is constitutive equation developer most typified in AIAA-type fluid dynamics by the turbulence modeler. Again, the same person could and has functioned in all three capacities (code builder, code user, and constitutive equation developer), but I agree with Blottner[11] that it is too much burden on one person.

Code Verification via Systematic Grid Convergence Testing

In this section, emphasis is given to convincing, rigorous code verification via systematic grid convergence testing. Reference 30 presented a general approach to rigorous code verification via systematic grid convergence, now called the method of manufactured solutions. This procedure is straightforward though somewhat tedious to apply and verifies all aspects of the code: formulation of the discrete equations (interior and boundary conditions) and their order of accuracy, the solution procedure, and the user instructions.

The basic idea is to include in the code a general source term $Q(x, y, z, t)$ and to use it to generate a nontrivial but known solution structure. Following the counsel of Polya[31] (only a fool starts at the beginning; a wise man starts at the end...), we first chose a continuum solution. We want one that is nontrivial but analytic and exercises all ordered derivatives in the error expansion and all terms, e.g., cross-derivative terms. For example, choose a solution involving the hyperbolic tangent function tanh. This solution also defines boundary conditions, to be applied in any (all) forms, i.e., Dirichlet, Neuman, Robin, etc. Then the solution is passed through the governing partial differential equations (Ref. 30 used symbolic manipulation) to give the production term $Q(x, y, z, t)$ that produces this solution. (This procedure is much easier and more general than looking for solutions to real problems; see additional references in Ref. 6.) We then monitor the numerical error as the grid is systematically refined. (Successive grid halving is not required, just refinement. Thorough iterative convergence is required.) Theoretically, values of $c = \text{error}/\Delta^p$ should become constant as the grid is refined for a uniformly pth-order method ("uniformly" implying at all points for all derivatives). When this systematic grid convergence test is verified, we have verified 1) any equation transformations used (e.g., nonorthogonal boundary fitted coordinates), 2) the order of the discretization, 3) its encoding, and 4) the solution procedure.

This technique was originally applied[30–33] to long Fortran code produced by artificial intelligence methods. The first versions of the code produced extremely long subroutines because the symbol manipulation code MACSYMA did not know the rules for intermediate expressions in the chain rule expansions and for the derivative of the inverse of a matrix function. (Steinberg[30] later taught these to MACSYMA.) The original three-dimensional nonorthogonal coordinate code contained about 1800 lines of dense Fortran. It would be impossible to check this code by reading the source, yet the procedure described verified it convincingly. (Surprisingly, round-off error was not a problem.)

If the code to be verified does not treat source terms, this capability must be added. Although often trivial, this can be difficult (and therefore an impediment to applying this method), notably for time accurate treatment of boundary conditions in approximate factorization algorithms. A related approach is to choose solution forms that are quadratic, for which a second-order code may (but not always) generate exact solutions on a finite grid. When this approach works, the verification can be accomplished without multiple grid solutions, but it still requires that the code treat a source term. An approach that does not require source terms is to manufacture the solution by manipulating spatially varying coefficients of the governing equations, e.g., the conductivity in a heat conduction equation.

There seems to be no chance for a really sweeping theorem proving correctness of CFD code by this technique in any general sense. However, like any analysis, it seems that there could be a useful theorem for a properly defined and limited scope. For codes like the one described earlier, treating only the well-behaved Poisson equation in general nonorthogonal coordinates, the exercise is compelling. I claim that this technique with this code (and a similar class of codes, which unfortunately I cannot define with sufficient mathematical theorem-like precision) is correct; i.e., the code is verified, beyond a reasonable doubt.

Grid Convergence Index

The Grid Convergence Index or GCI[27,34,35] presents a simple method for uniform reporting of grid convergence studies without any restriction to integer refinement, e.g., grid doubling. The GCI is based on generalized Richardson extrapolation involving comparison of discrete solutions at two different grid spacings. There may be good reasons[27,34,35] for not using the solution indicated by Richardson extrapolation, including lack of conservation. (Earlier, de Vahl Davis[36] pointed out the more fundamental problem that the extrapolated solution is "no longer internally consistent; the values of all the variables do not satisfy a system of finite difference approximations.") Richardson extrapolation is used herein only to obtain the error estimate, not to obtain a new solution.

A coarse grid Richardson error estimator approximates the error in a coarse grid solution, f_2, by comparing this solution to that of a fine grid, f_1, and is defined as

$$E_2^{\text{coarse}} = [r^p \varepsilon/(1 - r^p)] \tag{1}$$

whereas a fine grid Richardson error estimator approximates the error in a fine grid solution, f_1, by comparing this solution to that of a coarse grid, f_2, and is defined as

$$E_1^{\text{fine}} = [\varepsilon/(1 - r^p)] \tag{2}$$

where

$$\varepsilon = f_2 - f_1 \tag{3}$$

and where

f_2 = a coarse grid numerical solution obtained with grid spacing h_2

f_1 = a fine grid numerical solution obtained with grid spacing h_1

r = refinement factor between the coarse and fine grid [$r = (h_2/h_1) > 1$]

p = formal (or observed) order of accuracy of the algorithm.

It is neither necessary nor often desirable to use $r = 2$, or grid doubling; see Ref. 27 for discussion.

Accurate application of these generalized Richardson-based grid error estimators with theoretical p requires that the observed convergence rate equals the formal convergence rate. This implies that the leading order truncation error term in the error series truly dominates the error. In fact, these Richardson error estimators can be simply derived by considering a generic convergence curve for a pth-order approximation in the asymptotic range, without need for the Taylor series assumption of smoothness. This assumption would be needed only to evaluate coefficients from theory, which is not a requirement for grid convergence studies, nor is it even a practical possibility in a nontrivial CFD problem.

To account for the uncertainty in these generalized Richardson-based error estimates due to various factors, and to put all grid convergence studies on the same basis as grid doubling with a second-order method, we incorporate a safety factor into these estimators and define the GCI for coarse and fine grids as

$$\text{GCI}_2^{\text{coarse}} = F_s|E_2| \tag{4}$$

$$\text{GCI}_1^{\text{fine}} = F_s|E_1| \tag{5}$$

The term $F_s > 1$ can be interpreted as a safety factor because $F_s = 1$ gives GCI $= |E|$. That is, the error band reduces to the best estimate of the error, analogous to a 50% error band of experimental data. For a minimal two-grid convergence study, I recommended[27,34,35] a more conservative value of $F_s = 3$. This value also has the advantage of relating any grid convergence study (any r and p) to one with a grid doubling and a second-order method ($r = 2$, $p = 2$). I emphasize that the GCIs are not error estimators but are 3 (or F_s) times the error estimators, representing error bands in a loose statistical sense.

The motivation for using $F_s > 1$ is that $F_s = 1$ is analogous to a 50% error band on experimental data, which is not adequate. My originally recommended value[27,34,35] of $F_s = 3$ is conservative and relates the grid convergence study to one with a grid doubling with a second-order method. For many reasons[37,38] this is not unduly conservative when only two grids are used in the study. However, it is now clear that $F_s = 3$ is overly conservative for scrupulously performed grid convergence studies using three or more grid solutions to experimentally determine the observed order of convergence p, e.g., see the papers in Ref. 39. For such high-quality studies, a modest and more palatable value of $F_s = 1.25$ appears to be adequately conservative. However, for the more common two-grid study (often performed reluctantly, at the insistence of journal editors), I still recommend the value $F_s = 3$ for the sake of uniform reporting and adequate conservatism.

A recent application of the GCI to airfoil calculations is given by Lotz et al.,[40] which demonstrates the power of the method without the need for integer grid refinement and its application with solution-adaptive grids.

Sensitivity of Grid Convergence Testing

In our experience, this method of code verification via systematic grid convergence testing (whether or not the GCI is used) is remarkably sensitive in revealing code problems, as indicated by the following examples.

1) In verification tests of a commercial groundwater flow code, a first-order error in a single corner cell in a strongly elliptic problem caused the observed convergence to be first-order accurate.[12]

2) In verification tests of our SECO-FLOW-2D variable density groundwater flow code, first-order extrapolation for ghost cell values of only one quantity (aquifer thickness) along one boundary caused the observed convergence to be first-order accurate.[41]

3) In groundwater contaminant transport calculations (advection-diffusion + decay, retardation, and matrix diffusion), use of a plausible single-grid-block representation for a point source as the grid is refined introduces error in a finite volume (or block-centered finite difference) formulation. In this cell configuration, the cell faces align with the boundaries of the computational domain, and doubling the number of cells requires the location of the single cell representing the source to shift by $\Delta/2$. It is to be expected that the solution accuracy in the neighborhood of the source would be affected. But surprisingly, the accuracy of time-integrated discharge across boundaries far from the source was also degraded to first-order accuracy.[42]

4) The observed convergence rate of ostensibly second-order accurate turbulent boundary-layer codes[43] can be degraded, apparently by conditional statements limiting eddy viscosity and defining the boundary-layer edge.

5) Airfoil codes can exhibit the expected second-order convergence rates for lift and drag, but less for moment, possibly because of approximations involved in applying quasiperiodicity across cut-planes of a C-grid.

Esoteric Coding Errors

General CFD (or computational physics) codes (more general than the simple Poisson equation in nonorthogonal coordinates[30]) would be difficult to include in a theorem because of esoteric coding errors. The difficult aspects of the codes are not algebraic complexity (we convincingly verified 1800 lines of dense Fortran[30]). The more difficult and vexing problems come from option combinations and conditional differencing. Esoteric errors can arise because of nonlinear flux limiters like FCT, TVD, hybrid or type-dependent differencing, etc.[30] Other examples of error types that might escape detection during incomplete verification exercises are neglected REAL declarations that still produce correct results but only for integer values of some parameters[44,45] and near-conformal grid generation that does not exercise cross-derivative terms.[46,47]

Special Considerations for Turbulence Modeling

There are special considerations required for turbulence modeling and for other fields with multiple scales. Here, the code theoretical performance can be verified (within a tolerance) for a range of parameters but could fail in another range.

It is necessary to get the grid resolution into the asymptotic range to do grid convergence testing. Virtually any grid is in the asymptotic range for a simple Laplace equation. For any boundary-layer calculation, it is clear that the initial (coarse) grid must get some points into the boundary layer. For turbulence modeling without wall functions,[43,48] the grid must get some points into the wall layer. For turbulence modeling with wall functions, the grid should not get into the wall layer.[49] In my interpretation, effectively the wall functions should be viewed as an elaborate nonlinear boundary condition, and the grid convergence exercise should be done from the edge of the wall layer out. Similarly, for large eddy simulations (LES) as used in aerodynamic turbulence research and in atmospheric and ocean modeling with subgrid turbulence modeling, the grid convergence must not go to zero or else the Reynolds stresses will be counted twice, once from the full Navier–Stokes terms and again modeled from the LES terms. Also, the presence of any switching functions, such as length determinations for the Baldwin–Lomax turbulence model,[43] can easily corrupt second-order convergence rates.

Finally, the grid resolution requirements are much more demanding for turbulent boundary layers, just as laminar boundary layers are much more demanding than inviscid flows. For example, Claus and Vanka[50] found that 2.4 million nodes ($256 \times 96 \times 96$) did not demonstrate grid independence of the computed velocity and turbulence fields of crossflow jets.

Extraction of Observed Order from Grid Convergence Tests

If an exact solution is known or constructed (see the preceding text), it is straightforward to extract the order of convergence from results of a systematic grid convergence test using a minimum of two grid solutions. This serves to verify a code. However, it is also desirable to verify the order for an actual problem because the observed order of convergence depends on achieving the asymptotic range, which is problem dependent, and because the observed order may differ from the theoretical order, or from the order verified for a test case, for a variety of reasons. (See the discussion in Ref. 38.)

Blottner[11] and others use graphical means, plotting the error on log paper and extracting the order from the slope. This procedure requires evaluation of the error, which is generally not known. If the finest grid solution is taken to be the reference value (unfortunately, often called the exact value, which it obviously is not), then the observed order will be accurate only for those grids far from the finest, and the calculated order approaching the finest grid will be indeterminate. Blottner[11] improves on this by estimating the exact value by Richardson extrapolation (see also Ref. 48), but this procedure is somewhat ambiguous because the order is needed to perform the Richardson extrapolation.

If the grid refinement is performed with constant r (not necessarily $r = 2$), the observed order can be extracted directly from three grid solutions, without a need for estimating the exact solution, following de Vahl Davis.[36] With 1 being the finest grid in the present notation,

$$p = \ell n\left(\frac{f_3 - f_2}{f_2 - f_1}\right) \Big/ \ell n(r) \tag{6}$$

A generalization of this procedure, not restricted to constant r, is possible using the generalized theory of Richardson extrapolation.[27,34,35] Equation (19) of Ref. 27 (typo corrected) may be used to verify an assumed order p. (It is not necessary to use the GCI itself.) One calculates

$$\frac{1}{\alpha} = \frac{\mathrm{GCI}_{12}^{\mathrm{fine}}}{\mathrm{GCI}_{23}^{\mathrm{fine}}} \tag{7}$$

If $\alpha \approx r^p$, then p is the observed order. However, Eq. (7) requires r to be constant over the three grid set, and it cannot be used to calculate p directly because p is implicitly present in the GCIs. The more general procedure is to solve the equation

$$\frac{\varepsilon_{23}}{\left(r_{23}^p - 1\right)} = r_{12}^p \left[\frac{\varepsilon_{12}}{\left(r_{12}^p - 1\right)}\right] \tag{8}$$

for p. This is simple for r constant (not necessarily 2 or integer), giving

$$p = \ell n(\varepsilon_{23}/\varepsilon_{12}) \tag{9}$$

But if r is not constant during the grid refinement, Eq. (8) is transcendental in p. Usual solution techniques (e.g., Newton–Raphson) can apply, but one should allow for observed $p < 1$. This can happen even for simple problems at least locally[36] and in some cases the observed p is < 0 (unfortunately, behavior far away from asymptotic convergence can be nonmonotone). Also, $r \sim 2$ will be easier to solve than $r \sim 1+\delta$, and $r \gg 2$ is probably not of much interest. For well-behaved synthetic cases, simple substitution iteration with a relaxation factor $\omega \sim 0.5$ works well. With ρ = previous iterate for p, the iteration equation is

$$p = \omega\rho + (1-\omega)\frac{\ell n(\beta)}{\ell n(r_{12})} \tag{10a}$$

$$\beta = \frac{\left(r_{12}^{\rho}-1\right)}{\left(r_{23}^{\rho}-1\right)}\frac{\varepsilon_{23}}{\varepsilon_{12}} \tag{10b}$$

Note this form of the iteration gives the exact answer in one step for the case of $r = 2$.

Once p is known with some confidence, one may predict the next level of grid refinement r^* necessary to achieve a target accuracy, expressed as a target error estimate E_1 or GCI_1, call it GCI*. With GCI_{23} being the value from Eq. (2) for the previous two grids,

$$r^* = \sqrt[p]{\mathrm{GCI}^*/\mathrm{GCI}_{23}} \tag{11}$$

(This result, of course, depends only on the assumed definition of order of the discretization error, i.e., only on $c = \text{error}/\Delta^p$, and not on the GCI theory itself.)

The theoretical p may be directional, e.g., perhaps a boundary-layer code that is first order in the streamwise direction and second order otherwise. The preceding procedure may be applied either to the overall observed p (producing some observed value of p that is intermediate) or to each direction refined independently.

Cumulative Area Fraction Error Curves

Cumulative area fraction error curves introduced by Luettich and Westerink[51] present domain errors in a complete and meaningful way, plotting the fraction of the total domain that exceeds a particular error level against that error level. See also Refs. 37 and 52.

Surrogate Single-Grid Indexes

Many attempts have been made to estimate ordered numerical error using only a single solution on a single grid. (The original Richardson paper[53] in 1908 included examination of the difference between a low-order solution and a high-order solution on the same grid, which is obviously a low-ordered error estimator but is expensive.) More modern attempts try for a single grid solution with some inexpensive postprocessing, e.g., residuals of an FEM. These attempts are usually focused on directing solution adaptive grid generation, and they are usually successful for this purpose.[54] However, almost anything is successful for adaptation purposes, e.g., minimizing solution curvature or adapting to solution gradients (even though solution gradients cause no discretization error). Estimates of local errors usually are not what we want for verification of calculations; we need global errors.

Note that even lift and drag coefficients converge at different rates, so that a percentage error in one cannot naively be used for the other unless a quantitative correlation is established. Commonly, authors speak of a calculation being converged as though it had some general sense, when in fact convergence must be evaluated separately for each quantity of interest.

What is needed then is to establish a quantitative correlation between these local error estimates (or other functionals) and the errors of interest. If a reliable correlation can be established, then such a local inexpensive error estimate obtained on a single grid can be used as a surrogate for error estimates that would otherwise require multiple grid solutions.

Such a single-grid error index is provided by the energy imbalance method of Refs. 55 and 56. As noted earlier,[25] any nonconserved quantity can be used as grid convergence check. And we can always create a higher moment that is not conserved (because if all were conserved, the solution would be exact). Haworth et al.[55] use a kinetic energy (KE) equation. Note the importance of production and inflow terms, consistently evaluated. They set up the "KE equation imbalance" (like a residual), the evaluation of which gives an indication of lack of convergence. Because the limit is known without an exact solution, i.e., KE imbalance $\to 0$ as $\Delta \to 0$, it takes one less grid solution to establish the order of discretization error convergence by monitoring this functional than some term like a lift coefficient. That is, it takes three distinct grid solutions to establish the order of convergence when examining something like C_L or pressure at a point (or L_2 or L_∞ norms), but it requires only two grids to establish the order when examining KE imbalance or a similar term.

As noted, the problem is that KE imbalance or other conserved functionals are not often of intrinsic interest. We want to know error bands for C_L, for example. The utility of the single-grid evaluation can only be established by quantitative correlation with quantities of interest (e.g., C_L), and this requires complete convergence testing. (The same is true, I maintain, for the various single-grid solution approaches based on FEM theory.) The usefulness will be significant when such a correlation is established for a class of problems and when one can use this single-grid index as a surrogate for a suite of nearby problems, e.g., hundreds to thousands of Monte Carlo runs required for performance assessment of the WIPP (Waste Isolation Pilot Plant).[57]

The definitions used herein are sometimes at variance with other discussions,[58] notably in the present title subject of verification.

For a more complete discussion of the ideas in this paper, see Refs. 38 and 59.

Conclusion

The semantic distinctions involved in the general area of confidence building in CFD are important and worthwhile. Although the distinctions are sometimes arbitrary, it is worthwhile to try to maintain uniformity of terminology or at least to recognize the underlying conceptual distinctions and to define one's terms with appropriate precision.

My position is that we verify a code by convincingly demonstrating (if not proving as in a mathematical theorem) that the code can solve the equations right. (When done properly, the exercise also verifies the order of convergence of the code.) Then, we verify a calculation by convincingly demonstrating that the code has solved the equations right to a rationally estimated accuracy or error band. Neither of these exercises appeals to experimental data for their justification. Only in validation do we demonstrate that we have solved the right equations with an understood context of engineering or scientific accuracy.

Finally, techniques are already available to convincingly verify the numerical accuracy of CFD codes and calculations without undue stress on computer resources.

Acknowledgments

This work was partially supported by Sandia National Laboratories and the U.S. Department of Energy under Contract DE-AC04-76DP00789; M. G. Marietta and M. Fewell were the technical monitors. I am indebted to F. G. Blottner and an another anonymous reviewer for many improvements. D. C. Wilcox and K. Salari provided the fourth and fifth examples of the sensitivity of grid convergence testing.

References

[1]Roache, P. J., Ghia, K., and White, F., "Editorial Policy Statement on the Control of Numerical Accuracy," *Journal of Fluids Engineering*, Vol. 108, No. 1, 1986, p. 2.

[2]Freitas, C. J., "Editorial Policy Statement on the Control of Numerical Accuracy," *Journal of Fluids Engineering*, Vol. 115, No. 2, 1993, p. 339.

[3]Editorial Board, "Journal of Heat Transfer Editorial Policy Statement on Numerical Accuracy," *Journal of Heat Transfer*, Vol. 116, Nov. 1994, pp. 797, 798.

[4]AIAA, "Editorial Policy Statement on Numerical Accuracy and Experimental Uncertainty," *AIAA Journal*, Vol. 32, No. 1, 1994, p. 3.

[5]Gresho, P. M., and Taylor, C., "Editorial," *International Journal for Numerical Methods in Fluids*, Vol. 19, 1994, p. iii.

[6]Roache, P. J., "Need for Control of Numerical Accuracy," *Journal of Spacecraft and Rockets*, Vol. 27, No. 2, 1990, pp. 98–102; also AIAA Paper 89-1669, 1989.

[7]Horgan, J., "From Complexity to Perplexity," *Scientific American*, Vol. 272, No. 6, 1995, pp. 104–109.

[8]Konikow, L. F., and Bredehoeft, J. D., "Groundwater Models Cannot be Validated," *Advances in Water Resources*, Vol. 15, 1992, pp. 75–83.

[9]Oreskes, N., Shrader-Frechette, K., and Belitz, K., "Verification, Validation, and Confirmation of Numerical Models in the Earth Sciences," *Science*, Vol. 263, Feb. 1994, pp. 641–646.

[10]Boehm, B. W., *Software Engineering Economics*, Prentice–Hall, Englewood Cliffs, NJ, 1981.

[11]Blottner, F. G., "Accurate Navier–Stokes Results for the Hypersonic Flow over a Spherical Nosetip," *Journal of Spacecraft and Rockets*, Vol. 27, No. 2, 1990, pp. 113–122.

[12]Aeschliman, D. P., Oberkampf, W. L., and Blottner, F. G., "A Proposed Methodology for Computational Fluid Dynamics Code Verification, Calibration, and Validation," *Proceedings of the International Congress on Instrumentation in Aerospace Simulation Facilities*, 1995.

[13]Haynes, T. S., Reed, H. L., and Saric, W. S., "CFD Validation Issues in Transition Modeling," AIAA Paper 96-2051, 1996.

[14]Roache, P. J., Knupp, P. M., Steinberg, S., and Blaine, R. L., "Experience with Benchmark Test Cases for Groundwater Flow," *Benchmark Test Cases for Computational Fluid Dynamics*, edited by I. Celik and C. J. Freitas, ASME FED, Vol. 93, Book H00598, American Society of Mechanical Engineers, 1990, pp. 49–56.

[15]Mehta, U., "Computational Requirements for Hypersonic Flight Performance Estimates," AIAA Paper 89-1670, June 1989.

[16]Melnik, R., and Siclari, M., "An Overview of a Recent Industry Effort at CFD Code Validation," AIAA Paper 95-2229, 1995.

[17]Mehta, U., "Some Aspects of Uncertainty in Computational Fluid Dynamics Results," *Journal of Fluids Engineering*, Vol. 113, 1995, pp. 538–543.

[18]Mehta, U., "Guide to Credible Computational Fluid Dynamics Simulations," AIAA Paper 95-2225, 1995.

[19]Cosner, R. R., "CFD Validation Requirements for Technology Transition," AIAA Paper 95-2227, 1995.

[20]Melnik, R. E., Siclari, M. J., Marconi, F., Barger, T., and Verhoff, A., "An Overview of a Recent Industry Effort at CFD Code Certification," AIAA Paper 95-2229, 1995.

[21]Patankar, S. V., *Numerical Heat Transfer and Fluid Flow*, McGraw–Hill, New York, 1980.

[22]Ferziger, J. H., "Estimation and Reduction of Numerical Error," *Quantification of Uncertainty in Computational Fluid Dynamics*, edited by I. Celik, C. J. Chen, P. J. Roache, and G. Scheurer, ASME FED, Vol. 158, American Society of Mechanical Engineers, 1993, p. 18.

[23]Karniadakis, G. E., "Toward a Numerical Error Bar in CFD," *Journal of Fluids Engineering*, Vol. 117, No. 1, 1995, pp. 7–9.

[24]Vanka, P., "Comment on Toward a Numerical Error Bar in CFD," *Journal of Fluids Engineering*, Vol. 117, No. 1, 1995, pp. 9, 10.

[25]Roache, P. J., *Computational Fluid Dynamics*, Hermosa Publishers, Albuquerque, NM, 1972; revised printing, 1976; also *Fundamentals of Computational Fluid Dynamics*, 1998.

[26]Zingg, D. W., "Grid Studies for Thin-Layer Navier–Stokes Computations of Airfoil Flowfield," *AIAA Journal*, Vol. 30, No. 10, 1993, pp. 2561–2564.

[27]Roache, P. J., "Perspective: A Method for Uniform Reporting of Grid Refinement Studies," *Journal of Fluids Engineering*, Vol. 116, No. 3, 1994, pp. 405–413.

[28]Sudicky, E. A., and Frind, E. O., "Contaminant Transport in Fractured Porous Media: Analytical Solutions for a System of Parallel Fractures," *Water Resources Research*, Vol. 18, No. 6, 1982, pp. 1634–1642.

[29]Roache, P. J., "Response: To the Comments by Drs. W. Shyy and M. Sindir," *Journal of Fluids Engineering*, Vol. 116, No. 2, 1994, pp. 198, 199.

[30]Steinberg, S., and Roache, P. J., "Symbolic Manipulation and Computational Fluid Dynamics," *Journal of Computational Physics*, Vol. 57, No. 2, 1985, pp. 251–284.

[31]Polya, G., *How to Solve It; A New Aspect of Mathematical Method*, Princeton Univ. Press, Princeton, NJ, 1953.

[32]Roache, P. J., and Steinberg, S., "Symbolic Manipulation and Computational Fluid Dynamics," AIAA Paper 83-1952, 1983; also *AIAA Journal*, Vol. 22, No. 10, 1984, pp. 1390–1394.

[33]Steinberg, S., and Roache, P. J., "A Tool Kit of Symbolic Manipulation Programs for Variational Grid Generation," AIAA Paper 86-0241, 1986.

[34]Roache, P. J., "A Method for Uniform Reporting of Grid Refinement Studies," *Quantification of Uncertainty in Computational Fluid Dynamics*, edited by I. Celik, C. J. Chen, P. J. Roache, and G. Scheurer, ASME FED, Vol. 158, American Society of Mechanical Engineers, 1993, pp. 109–120.

[35]Roache, P. J., "A Method for Uniform Reporting of Grid Refinement Studies," *Proceedings of the AIAA 11th Computational Fluid Dynamics Conference*, Pt. 2, AIAA, Washington, DC, 1993, pp. 1057, 1058.

[36]de Vahl Davis, G., "Natural Convection of Air in a Square Cavity: A Benchmark Numerical Solution," *International Journal for Numerical Methods in Fluids*, Vol. 3, No. 3, 1983, pp. 249–264.

[37]Westerink, J. J., and Roache, P. J., "Issues in Convergence Studies in Geophysical Flow Computations," Joint JSME–ASME Fluid Mechanics Meeting, Session F137, 1995.

[38]Roache, P. J., "Quantification of Uncertainty in Computational Fluid Dynamics," *Annual Review of Fluid Mechanics*, Vol. 29, 1997, pp. 123–160.

[39]Johnson, R. W., and Hughes, E. D., eds., *Quantification of Uncertainty in Computational Fluids Dynamics—1995, Joint JSME-ASME Fluid Mechanics Meeting*, ASME FED, Vol. 213, American Society of Mechanical Engineers, 1995.

[40]Lotz, R. D., Thompson, B. E., Konings, C. A., and Davoudzadeh, F., "Numerical Uncertainties in Transonic Flow Calculations for Airfoils with Blunt Trailing Edges," *International Journal for Numerical Methods in Fluids*, Vol. 24, 1997, pp. 355–373.

[41]Roache, P. J., "The SECO Code Algorithms for Groundwater Flow and Transport," *Finite Elements in Fluids: New Trends and Applications, Part II, Proceedings VIII International Conference on Finite Elements in Fluids*, edited by K. Morgan, E. Onate, J. Periaux, J. Peraire, and O. C. Zienkiewicz, 1993, pp. 939–948.

[42]Salari, K., Blaine, R. L., Economy, K., and Roache, P. J., "Grid Resolution Studies of Radionuclide Transport in Fractured Porous Media," *Joint JSME-ASME Fluid Mechanics Meeting*, ASME FED, Vol. 213, American Society of Mechanical Engineers, 1995.

[43]Wilcox, D. C., *Turbulence Modeling for CFD*, DCW Industries, La Cañada, CA, 1993.

[44]Mueller, T. J., Hall, C. R., and Roache, P. J., "Influence of Initial Flow Direction on the Turbulent Base Pressure in Supersonic Axisymmetric Flow," *Journal of Spacecraft and Rockets*, Vol. 7, No. 12, 1970, pp. 1484–1488.

[45]Roache, P. J., "Base Drag Calculations in Supersonic Turbulent Axisymmetric Flows," *Journal of Spacecraft and Rockets*, Vol. 10, No. 4, 1973, pp. 285–287.

[46]Roache, P. J., "Semidirect/Marching Methods and Elliptic Grid Generation," *Proceedings of the Symposium on the Numerical Generation of Curvilinear Coordinate Systems and Use in the Numerical Solution of Partial Differential Equations*, edited by J. F. Thompson, 1982, pp. 727–737.

[47]Roache, P. J., *Elliptic Marching Methods and Domain Decomposition*, CRC Press, Boca Raton, FL, 1995.

[48]Shirazi, S. A., and Truman, C. R., "Evaluation of Algebraic Turbulence Models for PNS Predictions of Supersonic Flow Past a Sphere-Cone," *AIAA Journal*, Vol. 27, No. 5, 1989, pp. 560–568.

[49]Celik, I., and Zhang, W.-M., "Calculation of Numerical Uncertainty Using Richardson Extrapolation: Application to Some Simple Turbulent Flow Calculations," *Journal of Fluids Engineering*, Vol. 117, Sept. 1995, pp. 439–445.

[50]Claus, R. W., and Vanka, S. P., "Multigrid Calculations of a Jet in Crossflow," *Journal of Propulsion and Power*, Vol. 8, 1992, pp. 185–193.

[51]Luettich, R. A., and Westerink, J. J., "Continental Shelf Scale Convergence Studies with a Barotropic Tidal Model," *Quantitative Skill Assessment for Coastal Ocean Models*, edited by D. R. Lynch and A. M. Davies, AGU Press, 1995.

[52]Westerink, J. J., Luettich, R. A., and Muccino, J. C., "Modeling Tides in the Western North Atlantic Using Unstructured Graded Grids," *Tellus*, 46A, 1994, pp. 187–199.

[53]Richardson, L. F., "The Approximate Arithmetical Solution by Finite Differences of Physical Problems Involving Differential Equations, with an Application to the Stresses in a Masonry Dam," *Transactions of the Royal Society of London*, Series A, Vol. 210, 1908, pp. 307–357.

[54]Oden, J. T., Kennon, S. R., Tworzydlo, W. W., Bass, J. M., and Berry, C., "Progress on Adaptive h-p Finite Element Methods for the Incompressible Navier–Stokes Equations," *Computational Mechanics*, Vol. 11, 1993, pp. 421–432.

[55]Haworth, D. C., El Tahry, S. H., and Huebler, M. S., "A Global Approach to Error Estimation and Physical Diagnostics in Multidimensional Computational Fluid Dynamics," *International Journal for Numerical Methods in Fluids*, Vol. 17, 1993, pp. 75–97.

[56]Chang, S., and Haworth, D. C., "Adaptive Grid Refinement Using Cell-Level and Global Imbalances," *International Journal for Numerical Methods in Fluids*, Vol. 24, 1997, pp. 375–392.

[57]WIPP PA Department, *Annual Performance Assessment for the Waste Isolation Pilot Plant, Vol. 1, Preliminary Comparison with 40 CFR 191, Subpart B, Vol. 2: Technical Basis, Performance Assessment Department*, Sandia National Lab., Albuquerque, NM, 1992.

[58]Oberkampf, W. L., "A Proposed Framework for Computational Fluid Dynamics Code Calibration/Validation," AIAA Paper 94-2540, June 1994.

[59]Roache, P. J., *Verification and Validation in Computational Science and Engineering*, Hermosa Publishers, Albuquerque, NM, 1998.

J. Kallinderis
Associate Editor