Chapter 12
The Method of Manufactured Solutions for Code Verification

Patrick J. Roache

Abstract Verification of codes that numerically approximate solutions of partial differential equations consists in demonstrating that the code is free of coding errors and is capable, given sufficient discretization, of approaching exact mathematical solutions. This requires the evaluation of discretization errors using known benchmark solutions. The best benchmarks are exact analytical solutions with a sufficiently complex solution structure; they need not be physically realistic since verification is a purely mathematical exercise. The Method of Manufactured Solutions (MMS) provides a straightforward and general procedure for generating such solutions. For complex codes, the method utilizes symbolic manipulation, but here it is illustrated with simple examples. When used with systematic grid refinement studies, which are remarkably sensitive, MMS can produce robust code verifications with a strong completion point.

Keywords Manufactured solutions · Simulation · Benchmark · Verification · Turbulence · Convergence · Symbolic manipulation

12.1 Introduction

We are concerned in this chapter only with simulation models that are based on discretization of partial differential equations (PDEs). This covers most of classical physics, broadly defined, as well as some models in economics, ecological systems, and other disciplines of basic and applied science and engineering.

General code verification was defined rather tersely by the IEEE three decades ago (IEEE 1991) as "Formal proof of program correctness." This definition has stood the test of time, but arguably benefits from expanded description; e.g., see Roache (1998a, b, p. 26 ff.) A definition specific to PDE codes in the context of computational solid mechanics was given in ASME (2006, p. 23): "the process of

P. J. Roache (✉)
Consultant, 1215 Apache Dr., Socorro, NM 87801, USA
e-mail: hermosa@sdc.org

C. Beisbart and N. J. Saam (eds.), *Computer Simulation Validation*,
Simulation Foundations, Methods and Applications,
https://doi.org/10.1007/978-3-319-70766-2_12

determining that the numerical algorithms are correctly implemented in the computer code and of identifying errors in the software." See also ASME (2009), Oberkampf and Roy (2010), and Chaps. 2, 3, 5, and 11 in this volume. Generally, we find that such legalistic definitions tend to be sterile and/or inadequate, and that expanded descriptions are more useful. Basically, a code should do what the code manual says it does. See discussions throughout Chap. 2 of Roache (1998a, b).

For verification of PDE codes, we use a restricted definition of code verification, being concerned only with the ability of the code to produce mathematically accurate answers when sufficient discretization resolution is used. (This can contrast with computer science concepts of code verification that might include Quality Assurance issues that may have no effect on accuracy.) Determining this restricted sense of code correctness can only be accomplished by systematic discretization convergence tests using a "benchmark" solution which is preferably exact but at least reliable.

Once the verification of the code has been established, one can solve a specific problem which, if it is nontrivial, does not have an available exact solution. Verification of the computational *solution* then involves error *estimation*, since the benchmark solution is not known, whereas verification of the *code* involved error *evaluation* from a known benchmark solution. Both verifications (of code and solution) are purely mathematical activities, with no concern whatever for the accuracy of physical laws. That is the concern of validation, i.e., the agreement of the mathematics with observational science. In this view (see Chap. 27 by Roache in this volume), validation of computer simulations requires the three distinct activities referred to collectively as Verification and Validation (V&V): verification of the code, verification of the solution, and validation. For reasons both logical and practical, these activities should be performed in this order (Roache 2009; see also Chap. 42 by Beisbart in this volume).

The best benchmark solution is an exact analytical solution, i.e., a solution expressed in simple primitive functions like *sin*, *exp, tanh*, etc. Benchmark solutions involving infinite series are not desirable, typically being more numerical trouble to evaluate accurately than the PDE code itself (Roache 2009). It is not sufficient that the benchmark solution be exact; it is also necessary that the solution structure be sufficiently complex that all terms in the governing equation being tested are well exercised. Some early and misleading claims of accuracy of commercial codes for computational fluid dynamics (CFD) were based on comparisons with Poiseuille, Couette or Rayleigh problems, which do not even activate the advection terms. Many papers and reports have approached verification of codes in a haphazard and piecemeal way, comparing single-grid results for a few exact solutions on problems of greatly reduced complexity.

The Method of Manufactured Solutions (MMS) provides a systematic and general procedure for generating analytical solutions for code verification. The methodology provides for convincing, robust verification of a code via systematic discretization convergence testing. This procedure is straightforward though tedious to apply, and verifies all accuracy aspects of the code: formulation of the discrete equations (interior and boundary conditions) and their order of accuracy, the accuracy of the solution procedure, and the user instructions.

For early history and references, see (Roache 2002, 2009). The first systematic exposition of the method with application to multidimensional nonlinear problems appears to be (Roache and Steinberg 1984, Steinberg and Roache 1985) with expanded applications in Roache (2002, 2009). Acceptance was slow and misunderstanding was not uncommon, even by senior researchers. Now, MMS is regarded by many V&V specialists as the "gold standard" for PDE code verification, but still, it can be difficult to understand on first exposure. Based on experience of colleagues and myself, including teaching short courses, the misunderstanding seems due to the deceptive simplicity (elegance?) of the concept. Although applicable to high dimensional complex problems, the MMS concept is best described with simple examples in one space dimension (1D).

In what follows, Sect. 12.2 will first present the basic idea of MMS for generating exact benchmark solutions. Section 12.3 will illustrate this process using three simple examples. Then Sect. 12.4 will detail the application of benchmark MMS solutions to code verifications. The remaining sections present features and further examples of MMS.

12.2 Broad Description of MMS

The basic idea of the MMS procedure is to simply manufacture an exact solution, without being concerned about its physical realism. The "realism" or lack thereof has nothing to do with the mathematics, and verification is a purely mathematical exercise. In the original, most straightforward and most universally applicable version of the method, one simply includes in the code a general source term and uses it to generate a nontrivial but known solution structure. We follow the classic counsel of Polya (1957): *Only a fool starts at the beginning; the wise one starts at the end.*

We first pick a continuum solution. Interestingly enough, we can pick a solution almost independent of the code and the hosted equations (using a little prudence). That is, we can pick a solution, then use it to verify an incompressible Navier–Stokes code, a Darcy flow in porous media code, a heat conduction code, an electrode design code, a materials code, etc.

We want a solution that is nontrivial but analytic, and that exercises all ordered derivatives in the error expansion and all terms, e.g., cross-derivative terms. MMS can handle discontinuities (see below) but for this broad description, we consider smooth solutions. For example, chose a solution involving *tanh*. This solution also defines boundary conditions, to be applied in any (all) forms, i.e., Dirichlet, Neumann, Robin, etc. Then the solution is passed through the governing PDEs to give a source term that produces this solution. (This description sounds circular, which relates to difficulties with acceptance.)

12.3 Three Example Problems in MMS

To emphasize the generality of the concept, we pick the first example solution *before we specify the governing equations*. Then, we will use this same solution for two different problems, i.e., sets of governing PDEs and boundary conditions. The chosen solution *U(t,x)* is the following.

$$U(t,x) = A + \sin(B),\ B = x - Ct \tag{12.1}$$

12.3.1 Example 1

First, we apply this 1D transient solution to the nonlinear Burgers equation. (This equation is often taken as a model for CFD algorithm development but it is adequate to describe MMS for a wide range of PDE problems.)

$$u_t = -uu_x + \alpha u_{xx} \tag{12.2}$$

Incidentally, the specified solution *U(t,x)* is the exact solution for the constant velocity advection equation $u_t = -Cu_x$ with boundary condition $u(t,0) = A + sin(-Ct)$, so for high Reynolds number problems (small α) it may look "realistic" in some sense, but it is not a solution to our governing Eq. (12.2), and its "realism" or lack thereof is irrelevant to the task of code verification.

We determine the source term *Q(t,x)* which, when added to the Burgers equation for *u(t,x)*, produces the solution *u(t,x) = U(t,x)*. We write the Burgers equation as an operator (nonlinear) of *u*,

$$L(u) \equiv u_t + uu_x - \alpha\, u_{xx} = 0 \tag{12.3}$$

Then, we evaluate the *Q* that produces *U* by operating on *U* with *L*.

$$\begin{aligned} Q(t,x) &= L(U(t,x)) \\ &= \partial U/\partial t + U\partial U/\partial x - \alpha\, \partial^2 U/\partial x^2 \end{aligned} \tag{12.4}$$

By elementary operations on the manufactured solution *U(t,x)* stated in Eq. (12.1), we obtain

$$Q(t,x) = -Ccos(B) + [A + sin(B)]cos(B) + \alpha sin(B) \tag{12.5}$$

If we now solve the modified equation

$$L(u) \equiv u_t + uu_x - \alpha\, u_{xx} = Q(t,x) \tag{12.6}$$

$$u_t = -uu_x + \alpha\, u_{xx} + Q(t,x) \tag{12.7}$$

with compatible initial and boundary conditions, the exact solution will be *U(t,x)* given by Eq. (12.1).

The initial conditions are obviously just *u(0,x) = U(0,x)* everywhere. The boundary conditions are determined from the manufactured solution *U(t,x)* of Eq. (12.1). Note that, we have not yet even specified the domain of the solution. If we want to consider the usual model $0 \leq x \leq 1$ or something like it, the same solution Eq. (12.1) applies, but of course, the boundary values are determined at the corresponding locations in x. Note also that we have not yet even specified the *type* of boundary condition. This aspect of the methodology has often caused confusion. Everyone knows that different boundary conditions on a PDE produce different solutions; not everyone recognizes immediately that the same solution *U(t,x)* can be produced by more than one set of boundary condition *types*. The following combinations of inflow (left boundary, e.g., $x = 0$) or outflow (e.g., $x = 1$) boundary conditions will produce the same solution *U(t,x)* over the domain.

Dirichlet—Dirichlet:

$$u(t,0) = U(t,0) = A + sin(-Ct),\ u(t,1) = U(t,1) = A + sin(1 - Ct) \tag{12.8}$$

Dirichlet—Outflow Gradient (Neumann):

$$u(t,0) = A + sin(-Ct),\ \partial u \big/ \partial x(t,1) = cos(1 - Ct) \tag{12.9}$$

Robin (mixed)—Outflow Gradient (Neumann):

The Robin boundary condition, $F = au + bu_x = c$ where a, b, and c are constants, is to be applied as a time-dependent condition at the left boundary, so $F(t,0) = c$.

$$\begin{aligned} &a\,u + b\,u_x = c\ applied\ at\ (t,0) \rightarrow \\ &\qquad \text{given}\ a\ \text{and}\ b,\ \text{select}\ c\ = a[A + sin(-Ct)] + b\,cos(-Ct) \\ &\partial u \big/ \partial x(t,1) = cos(1 - Ct) \end{aligned} \tag{12.10}$$

For this time-dependent solution, the boundary values are time-dependent. It also will be possible to manufacture time-dependent solutions with steady boundary values, if required by the code.

12.3.2 Example 2

To further clarify the concept, we now apply the same solution to a different problem, choosing as the new governing PDE a Burgers-like equation that might be a candidate for a 1D turbulence formulation based on the mixing-length concept.

$$u_t = -uu_x + \alpha u_{xx} + 2\lambda\left[x(u_x)^2 + x^2 u_{xx}\right] \tag{12.11}$$

Writing the mixing-length model equation as a nonlinear operator of u,

$$L(u) \equiv u_t + uu_x - \alpha\, u_{xx} - 2\lambda[x(u_x)^2 + x^2 u_{xx}] = 0 \tag{12.12}$$

we evaluate the Q_m that produces U by operating on U with $L_m = L$ from (12).

$$\begin{aligned} Qm(t,x) &= Lm(U(t,x)) \\ &= \partial U/\partial t + U\partial U/\partial x - \alpha\partial^2 U/\partial x^2 - 2\lambda\,[x(\partial U/\partial x)^2 + x^2\partial^2 U/\partial x^2] \end{aligned} \tag{12.13}$$

By elementary operations on the same manufactured solution *U(t,x)* stated in Eq. (12.1),we obtain

$$\begin{aligned} Q_m(t,x) = &- C\,cos(B) + [A + sin(B)]cos(B) + \alpha\, sin(B) \\ &- 2\lambda\left[x\,cos^2(B) - x^2 sin(B)\right] \end{aligned} \tag{12.14}$$

If we now solve the modified model equation

$$Lm(u) \equiv u_t + uu_x - \alpha\, u_{xx} - 2\lambda[x(u_x)^2 + x^2 u_{xx}] = Qm(t,x) \tag{12.15}$$

$$u_t = -uu_x + \alpha\, u_{xx} + 2\lambda[x(u_x)^2 + x^2 u_{xx}] + Qm(t,x) \tag{12.16}$$

with compatible initial and boundary conditions, the exact solution for this "turbulent" problem again will be *U(t,x)* given by Eq. (12.1), as it was for the previous "laminar" problem.

The same initial and boundary conditions and boundary values from the previous problem can apply, since these are determined from the solution, not from the governing PDE nor Q.

12.3.3 Example 3

We have shown how the same solution can be used as the exact solution to verify two different codes with different governing equations, with different source terms being created to manufacture the same solution. A third example will demonstrate the arbitrariness of the solution form. Rather than the somewhat realistic solution to a constant velocity advection equation given by Eq. (12.1), we consider the "unrealistic" but equally valuable solution as follows.

$$U_e(t,x) = \sin(t)\, e^x \tag{12.17}$$

Following the same procedure for the Burgers Eq. (12.2), we evaluate the terms in Eq. (12.4) from the solution U_e of Eq. (12.17) and obtain

$$Q_e(t, x) = \cos(t)e^x + [\sin(t)e^x]^2 - \alpha \sin(t)e^x \tag{12.18}$$

(arranged for readability rather than compactness). This, when added to Eq. (12.2), produces the manufactured solution Eq. (12.17) when compatible initial and boundary conditions are evaluated from Eq. (12.17).

12.3.4 Complex Problems

MMS is applicable to complex nonlinear systems of equations, such as full Navier Stokes in general non-orthogonal coordinates, provided that the code is capable (or modifiable) to treat source terms in each PDE. MMS has been used in finite element codes both at the global solution level and at the element level (basis functions). To test periodic boundary conditions, one simply chooses a periodic function for the MMS solution.

12.4 Application to Code Verification

Once a nontrivial exact analytic solution has been generated, by MMS or perhaps another method, the solution is now used to verify a code by performing systematic discretization convergence tests (usually, grid convergence tests) and monitoring the convergence as $\Delta \to 0$, where Δ is a measure of discretization: Δx, Δt in a finite difference (FDM) or finite volume (FVM) code, element size in a finite element (FEM) code, etc. The procedure has been described in Chap. 11 by Rider in this volume; also Roache (1998a, b, 2009), Oberkampf and Roy (2010).

The fundamental concept "order of convergence" is based on behavior of the error of the discrete solution. There are various measures of error, but in some sense, we are always referring to the difference between the discrete solution $f(\Delta)$ (or a functional of the solution, such as drag coefficient) and the exact continuum solution,

$$E = f(\Delta) - f_{exact} \tag{12.19}$$

The most fundamental requirement for *code verification* is that $E \to 0$ as $\Delta \to 0$. In addition, we like to verify not only the *fact* of convergence but the *order* of convergence, ideally estimated a *priori* by analysis of the discretization methods used. By definition, for an order p method and for a well-behaved problem (exceptions are discussed in Roache 2009, Chaps. 6 and 8), the error in the solution E asymptotically as $\Delta \to 0$ will be proportional to Δ^p. This terminology applies to every mathemat-

ically consistent methodology: FDM, FVM, FEM, block spectral, pseudo-spectral, vortex-in-cell, etc., regardless of solution smoothness. Thus,

$$E = f(\Delta) - f_{exact} = C\,\Delta^p + H.O.T. \tag{12.20}$$

where HOT are higher order terms. We then monitor the numerical error as the grid is systematically refined. Thorough iterative convergence is required (see below). Successive grid halving is not required, just refinement. Theoretically (from Eq. 12.20), values of C= $E/\Delta^{\,p}$ should become constant as the grid is refined for a uniformly *p-th* order method ("uniformly" implying at all points for all derivatives). Formulaic details of the calculation of observed p from grid convergence testing and many examples are given in Roache (2009), Oberkampf and Roy (2010). If observed p is not ~ theoretical p, this may indicate a coding error, or it may indicate a limitation of the approximations in the analysis for theoretical p. In either case, the code is still useable and would be claimed as "verified" at the observed p. Confidence is greatly enhanced if observed p ~ theoretical p.

Roy (2001), Roy et al. (2000) showed how to treat mixed-order convergence, a long-standing and practical difficulty in grid convergence studies. Mixed-order behavior can arise from the use of first-order discretization for advection and second order for diffusion, or from the first-order convergence rate of nominally second-order methods applied to discontinuities. The procedure involves another grid level to evaluate *two* leading coefficients in the error expansion. Especially important, the papers demonstrate how non-monotonic convergence occurs from mixed-order methods in the non-asymptotic range without blaming nonlinearity. MMS can verify such mixed-order convergence.

Inadequate iterative convergence produces false-negative evaluations of observed p. The extrapolation implicit in the order calculation amplifies machine round-off errors, so the iteration error control is more demanding for evaluation of p than for the PDE solution itself. Unfortunately, a priori specifications of iterative convergence criteria (e.g., maximum allowable change of some solution metric over one iteration divided by the iteration relaxation parameter) are not reliable. The recommended procedure is to test the sensitivity of the code verification results (notably observed p) to the iterative convergence stopping criteria. Note that this difficulty is not specific to MMS but occurs with any calculation of observed p; in fact, widely chosen MMS solutions are less vulnerable than most classical solutions, as noted above. Also, note that (as many V&V specialists have warned), the default iteration stopping criteria used in commercial CFD codes are often highly inadequate.

This verification procedure detects all ordered errors *E,* i.e., $E \rightarrow 0$ asymptotically as $\Delta \rightarrow 0$. It will not detect coding mistakes that do not affect the answer obtained, e.g., mistakes in an iterative solution routine which affect only the iterative convergence rate. In the present view, these mistakes are not considered as code verification issues, since they affect only code efficiency, not accuracy. Note that such efficiency issues should not be a concern to regulatory agencies. Other esoteric mistakes that are difficult to detect are described in (Roache 2009, Chap. 8; Knupp and Salari 2003).

The procedure does not evaluate the adequacy of non-ordered approximations, e.g., distance to an outflow boundary, distance to an outer (wind tunnel wall-like) boundary, etc. The errors of such approximations (which, I claim, are not inherently "numerical") do not vanish as $\Delta \rightarrow 0$, hence are "non-ordered modeling approximations." The adequacy of these approximations must be assessed by sensitivity tests which may be described as "justification" exercises (Roache 2009).

When this systematic grid convergence test is verified for all point-by-point values, we have verified

- input routines
- any equation transformations (e.g., boundary fitted coordinates),
- the order of the discretization,
- the encoding of the discretization, and
- the accuracy (but not efficiency) of the matrix solution procedure.

This MMS technique was originally applied in Roache and Steinberg (1984), Steinberg and Roache (1985) to long Fortran code produced by Symbolic Manipulation methods. The original 3D non-orthogonal coordinate code contained about 1800 lines of dense Fortran. It would be impossible to check this by reading the source code, yet the MMS procedure verified the code convincingly. Round-off error was not a problem.

The technique of code verification by monitoring grid convergence is extremely powerful. Upon initial exposure to the technique, analysts are often negative about the method because they intuit that it cannot be sensitive enough to pick up subtle errors. After exposure to numerous examples, if they remain negative it is usually because the method is *excessively* sensitive, revealing minor inconsistencies such as first-order discretizations at a single boundary point in an elliptic problem that effects the *size* of the error very little (as correctly intuited) but still reduces the asymptotic rate of convergence to first *order* for the entire solution. For examples, see Roache (2009).

The fact that the MMS solution may bear no relation to any physical problem does not affect the rigor of the accuracy verification of codes. The only important point is that the solution (manufactured or otherwise) be nontrivial: it should exercise all the terms in the error expansion. The algebraic complexity may be something of a difficulty, but it is not insurmountable, and in practice has been easily handled using Symbolic Manipulation (SM) software packages. Using the source code writing capability of SM software, it is not even necessary for the analyst to look at the form of Q. Rather, the specification of the solution (e.g., *tanh* function) to the SM software results in some complicated analytical expression that can be directly converted by the SM software to a source code segment, which is then readily emplaced in a source code module (subroutine, function, etc.) that then is called in the code verification procedure. This "emplacement" can be performed by hand by the analyst without actually reading the complicated source code expressions, or can itself be automated in the SM software.

MMS has been applied successfully to nonlinear systems of equations, with separate Q's generated for each equation. Both steady (stationary) and unsteady man-

ufactured solutions may be formulated. Nonlinearity is an issue only because of uniqueness questions; the source term complexity may be worse because of nonlinearity, but managing that is the job of the SM software. Nonuniqueness conceivably could be an issue because the code might converge to another legitimate solution other than the MMS solution, producing a false-negative code verification. In much experience, nonuniqueness has never been an issue.

In Steinberg and Roache (1985), we applied the procedure to the coupled nonlinear (quasi-linear) PDEs of an elliptic grid generation method for non-orthogonal coordinates; the MMS solution was a 3D analytical coordinate transformation or parametrization. All operations for source code were performed by SM, including development of Euler–Lagrange equations for variational grid generation and all discretizations (Steinberg and Roache 1986a, b, 1992).

Note that the MMS solution should be generated in the original ("physical space") coordinates *(x,y,z,t)*. Then the same solution can be used directly with various non-orthogonal grids or coordinate transformations.

MMS (in this basic Q form) requires that the code being verified must include accurate treatment of source terms. Many codes, including the most popular modern commercial and open-source PDE codes, are built with source terms included, and many algorithms allow trivial extension to include Q's. However, directionally split algorithms (e.g., Roache 1998b) involve complexities at boundaries, especially for non-orthogonal coordinates.

Also see Roache (2009) for the following topics: early applications of MMS concepts, discussions and examples of mixed first- and second-order differencing, small parameter (high Reynolds number) difficulties, economics of dimensionality, applications of MMS to 3D grid generation codes, effects of strong and inappropriate coordinate stretching, debugging with MMS, examples of many manufactured or otherwise contrived analytical solutions in the literature, approximate but highly accurate solutions (often obtained by perturbation methods) that can also be utilized in code verification, special considerations required for turbulence modeling and other problems with multiple scales, example of MMS code verification with a 3D grid-tracked moving free surface, code robustness, examples of the remarkable sensitivity of code verification via systematic grid convergence testing, and several methodologies for verification of solutions including the Grid Convergence Index (see also Chap. 11 by Rider in this volume).

12.5 Features and Examples of MMS Code Verification

12.5.1 Radiation Transport Codes

Pautz (2001) presented his experience applying MMS to a radiation transport code that uses 3D tetrahedral elements in space and discrete ordinates in the angular discretization. The author discovered coding mistakes in input routines and in discretiza-

tion of certain boundary data. Second order convergence for norms and third-order convergence for average scalar flux were verified. A subtle aspect revealed is the requirement for consistent finite element weighting on the MMS source term, which is now a recognized issue. Based on the earlier 1D analysis in the literature, it was expected that all the examined quantities would exhibit third-order convergence, but the results of the MMS procedure demonstrated only second-order convergence for the norms in multidimensions.

Blackwell et al. (2009) applied MMS to enclosure radiation, verifying their non-rigorous theoretical analysis that indicated $p = 2$ in contrast to another analysis that indicated $p = 3$.

12.5.2 Nonhomogeneous and Nonlinear Boundary Conditions

An arbitrary MMS solution may have nonhomogeneous boundary conditions, e.g., $\partial u/\partial x \neq 0$. To use such manufactured solutions, the code would require the capability of treating boundaries with $\partial u/\partial x \neq 0$. This might be inconvenient, e.g., some codes have hard-wired no-slip conditions at a wall with $u = 0$, or $\partial u/\partial x = 0$. Rather than modify the code, some thought will produce MMS solutions with homogeneous boundary values. Fortunately, modern commercial and open-source PDE codes have this capability for general treatment of boundary conditions, which is also the feature that facilitates validation; see Roache (2004, 2009) and Chap. 27 by Roache in this volume.

The so-called "radiation" outflow conditions are usually linear and are already covered by the previous discussion. Nonlinear boundary conditions, e.g., simple vortex conditions at outflow, or true (physical) heat transfer radiation boundary conditions, are possible. It may be possible to select an MMS solution that meets the nonlinear boundary condition; otherwise, a source term would need to be used in the nonlinear boundary equations.

12.5.3 Shocks, Partitioning, and "Glass-Box" Verification

Shock solutions are treatable by the MMS, with additional considerations. The simplest approach is to verify the shock-capturing algorithms separately on inviscid benchmark problems such as oblique shock solutions, provided that shock curvature is not viewed as a major question. If it is, one may use attached curved shock solutions obtained by the method of characteristics and/or detached bow shock solutions obtained by the classical inverse method. Any shock-capturing algorithm based purely on geometric limiters will be oblivious to the source terms and should work without modification.

The assumption involved in this approach is that the option matrix of the code can be *partitioned* (Roache 2009). That is, the verification of the shock-capturing algorithm and coding will not be affected by later inclusion of viscous terms, boundary conditions, etc. Other option-partitioning assumptions will occur to the reader, such as: separated verification of a direct banded Gaussian elimination routine in a FEM code; verification of shock-capturing algorithms separate from nonideal gas effects; radioactive decay option (which is dimensionless) verified separately from the spatial discretization of flow equations. This partitioning approach requires the "black-box" verification philosophy to be modified to a "glass-box" (Oberkampf and Trucano 2002) in which some knowledge of code structure is required to justify the approach. Thus it will be more difficult to convince reviewers, editors, contract monitors, regulators, stakeholders, etc., that the approach is justified. The work savings can be enormous, of course, avoiding the factorial increase of complexity inherent in option combinations.

12.5.4 Shocks, Multiphase Flows, and Discontinuous Properties

Without using code partitioning, J. Powers and colleagues (Grismer and Powers 1996; see also Roache 2009 for additional references) pioneered convincing code verification for flows with shock waves. The benchmark solutions may involve asymptotic approximations in geometry and/or Mach number M, e.g., an analysis neglecting terms of order $\varepsilon = 1/M^2$. This approximation can be made very accurate by choosing high M, say $M \sim 20$. Note again the distinction of mathematics versus science; it is not a concern that the code being tested might be built on perfect gas assumptions that are not valid at such high M. This does not affect the mathematics of code verification; the code would not be applied at such high M when accuracy of the physics becomes important, during validation.

Woods and Starkey (2015) applied MMS to shocks and other discontinuities using an "integrative MMS approach" (contrasted to "differential MMS" herein) based on "intelligent subdivision of the integration domains" to obtain a rigorous, one-step verification procedure for shock-capturing codes.

Brady et al. (2012) applied MMS to multiphase flows which necessitate discontinuous properties at the interface, where careful evaluation of source terms is required. They also offer additional guidelines to help locate coding mistakes. MMS for multiphase flows were also considered by Choudhary et al. (2014).

Grier et al. (2014, 2015) treated discontinuous MMS solutions, focusing on numerical integration techniques to address the problem of evaluating source terms consistently in finite volume methods. FVM do not store solution values at the center of the cell but rather integrated average values, which will converge more slowly than expected to the MMS point values unless special care is taken in the integration; the discrepancy is aggravated by discontinuous MMS solutions. (Alternately, one might

consider post-processing the MMS solution to produce cell integrated average values for direct comparison, using methods consistent with the FVM code. However, this would add another layer of processing, the details of which would depend on the FVM solution code algorithm.)

Appendix A of ASME (2009) contains an MMS heat conduction problem with discontinuous step change in conductivity and contact resistance.

12.5.5 Verification of Boundary Conditions

Bond et al. (2007) presented an exemplary study applying MMS to CFD code verification of boundary conditions, including insightful observations. The FEM code being verified solves Euler, Navier–Stokes, and RANS equations on skewed, nonuniform, unstructured 3D meshes. Particular emphasis was placed on verification of numerical boundary conditions: slip, no-slip (adiabatic and isothermal), and outflow (subsonic, supersonic, and mixed), and on code segments that calculate solution gradients, a nontrivial issue in hexahedral grids with high aspect ratios near boundaries. The more demanding $L\infty$ norm was used and recommended, as well as the usual L_1 and L_2 norms. Among many interesting results, one provided a particular caution regarding precision issues. The symbolic manipulation software used to generate source functions writes source code in double precision but with only single precision constants, which later corrupted the initial verification exercise. The authors recommended an additional criterion for claiming verification of double-precision accuracy; the relative errors should be smaller than the single precision limit. Another caution involves orientation of the outflow boundary in supersonic flow along a constant pressure surface, which might permit certain coding errors to go undetected. This difficulty arose due to an ambitious approach of building boundary condition values into the MMS solution, rather than treating them crudely with the source term. Especially noteworthy was the success of MMS is disclosing a weakness of the solution algorithm in regard to the partitioning of multiprocessors. The paper is also valuable for presenting anecdotal debugging history, rather than a simple "pass" evaluation.

Choudhary et al. (2016) also focused on MMS verification of various important boundary conditions for both compressible and incompressible CFD codes.

12.5.6 Unsteady Flows and Divergence-Free MMS

An illustration of MMS applied to unsteady flows was given by Eça and Hoekstra (2007b). For the 2D laminar flows, a general formulation was developed that allowed an analyst to specify an arbitrary continuous function that is incorporated into an analytical form for velocities which satisfy the incompressible continuity constraint (divergence-free) exactly. Likewise, nonslip and impermeability conditions are met exactly by the MMS. Two time dependencies were considered: an exponen-

tially decaying solution and a periodic solution. The exercise verified the code, and additionally shed light iteration error.

Choudhary et al. (2016) also gave special attention to MMS solutions which identically satisfy the divergence-free velocity field for incompressible flows, and to curved boundaries.

12.5.7 Variable Density Flows; Combustion

Shunn et al. (2012a, b) used MMS for variable density PDE codes applicable to combustion problems. Issues included use of tabulated state properties and effects of sub-iterations in the time advancement, especially for problematical time-splitting methods.

12.6 Attributes of MMS Code Verification

12.6.1 Two Multidimensional Aspects

In the first 1D example problem (Sect. 12.3.1), we noted that the MMS solution, since it is analytic, can be applied over any range of the dependent spatial variable x. This feature extends to multidimensions, e.g., the same multidimensional analytic solution could be applied to flow problems of a rectangular cavity, a backstep, a wing, etc.

Also, multidimensional problems might require a little more thought to assure that all terms of the governing equations are exercised. For example, a manufactured solution of form $U(t,x,y) = F1(t) + F2(x) + F3(y)$ will not be adequate to exercise governing equations containing cross-derivative terms such as $\partial^2 u/\partial x \partial y$ since these are identically zero no matter how complex are the F's.

12.6.2 Blind Study

Salari and Knupp (2000) exercised MMS in a blind study, in which one author (Knupp) deliberately introduced errors into a CFD code previously developed and verified by the other (Salari). Then the code author tested the sabotaged code with the MMS. This exercise was not performed on merely model problems, but on a full time-dependent, compressible and incompressible, Navier–Stokes code with plenty of options. In all, 21 cases were studied, including one "placebo" (no mistake introduced) and several that involved something other than the solution (e.g., wrong time step, post-processing errors). Several formal mistakes (not order-of-convergence errors) went undetected, as expected (Roache 2002, 2009). All ten of the code errors that would affect accuracy were successfully detected, as well as several less serious mistakes.

12.6.3 Burden of MMS and Option Combinations

An experienced reviewer (Rider 2018) has stated that MMS puts a rather large burden on the code development teams, and that the source terms for MMS are difficult, prone to error, and need a high degree of software quality work and extensive debugging to produce reliable results.

I acknowledge this burden. In my own experience, the burden first involves some up-front work of becoming adept at using Symbolic Manipulation software. Once achieved, this development and training time can be amortized over verifications of many codes. In my own experience, the indictment of "prone to error" applies more to traditional methods. And it is true that producing a reliable and general MMS solution that exercises all the relevant terms certainly involves more work than coding an already developed single traditional solution, but possibly not if the simplified solution must be developed. Also, if one considers the suite of traditional problems usually required, then the amount of work may be less using a single MMS solution.

This claim is especially justifiable when one considers the curse of large numbers of code option combinations, discussed in Roache (2009). Suppose the non-separable option combinations number 100. A single MMS solution could easily replace a suite of 10 highly simplified classical solutions, reducing the required number of expensive grid convergence calculations by an order of magnitude, from 1000 to 100.

It is my opinion that the MMS approach is especially useful, reducing bookkeeping and total workload, when applied to extensive option combinations during *regression verification* of code modifications. (In major computational research environments, routine regression code verification activities are sometimes performed daily.)

12.6.4 Code Verification for Commercial Codes

Since code verification should be accomplished by the code developer, a question arises. Should a user assume that a commercial, open-source, or government code is verified? Roy (2015) reassessed previous misgivings by several V&V specialists and reached the same pessimistic evaluation: generally, the answer is no. This should be surprising since, as Roy noted, code verification is arguably the most mature subtopic in V&V; the main code verification techniques have been around for decades. Even if the vendor has published code verification for option combinations of interest, the user is strongly advised to scrutinize the results carefully for details and well founded conclusions.

12.6.5 Code Verification with a Strong Completion Point

Simple problems (even trivial problems) often serve a purpose during code development, and the results are often considered as partial verification. But apparently, it is not widely recognized that, once a code (for a specific set of code option combinations) has been convincingly verified on a complex problem that exercises all terms in the governing equations, it is nearly pointless to continue verifying the code on simpler problems. I say "nearly" because the exercises still have some value as *confirmation* exercises (Roache 2009, Chap. 1). MMS provides a robust code verification and terminates. A code user who performs confirmations gains confidence in the code and in their ability to set up the code and interpret the results. Such code confirmation exercises are valuable as part of user training but should not be confused with robust code verification. Similarly, we recognize that simple classical problems (e.g., 1D linear wave propagation) are useful in algorithm development, in exploring algorithm and code characteristics, and in comparing the performance of different codes. In fact, these classical problems are *more useful than MMS* for these purposes, since the general MMS solutions are typically unrealistic and opaque. But these comparison exercises, though valuable, should not be confused with robust code verification. These simple problems are complementary to the MMS approach, but if the comparison is taken as "partial verification" this leads to unending activity and invites criticism of the basic concept and legitimacy of code verification.

"Code verification is *not* an ongoing exercise. Verification, as we have said, is an exercise in mathematics, not science. When one proves a theorem, the work is completed. Proving the formula for solution of a quadratic equation is not ongoing work. This is not to say that one could not have made an error in the proof of a theorem, nor that confirmation exercises… are not valuable in confidence-building. It is to say that code verification is a mathematical activity that in principle comes to a conclusion, e.g., a code is or is not 2nd-order accurate." (Roache 1998a, b, p. 28) For an alternative view on code verification without a strong completion point, see discussion in (Roache 2002, 2009).

12.6.6 Proof?

Does such thorough code verification deserve the term *proof*? This is another semantic question whose answer depends on the community context. Logicians, philosophers and pure mathematicians clearly view "proof" differently from scientists and engineers, with an often other-worldly standard. For example, Fermat's Last Theorem is easily demonstrable, but do such exercises constitute *proof*? Certainly not to a mathematician. Since some philosophers maintain that it is not possible even in principle to prove Newton's laws of gravity, they are not likely to accept the notion of proof of correctness of a complex code.

The notion of *proof* is at the heart of very important criticisms, not just of the subject MMS, but of the concepts of code verification and especially *certification* for controversial public policy projects (Roache 2009). One might agree with philosophers who maintain it is not possible to prove Newton's Laws, but would one be willing to cancel a public policy project (e.g., nuclear waste disposal) because the modeling used Newton's Laws? Presumably not, but stakeholders are willing to cancel such projects under the guise of unprovability of code correctness. Great harm is done when these standards for proof of philosophers, mathematicians or logicians are applied to down-to-earth science and engineering projects. If we accept such out-of-context standards then we cannot do anything, literally. For example, we have no proof of convergence for realistic systems because the Lax Equivalence theorem only holds for linear systems (Roache 1998b).

The word *proof* is itself a technical term, with different appropriate standards in logic, pure mathematics, applied mathematics, engineering, criminal law versus torts versus civil law (consider "beyond a reasonable doubt"), etc. The first definition in one dictionary for *proof* is "The evidence or argument that compels the mind to accept an assertion as true." In this sense, if not in a mathematical sense, one could claim that MMS can provide proof of code verification. I am unhesitating in claiming "convincing demonstration" and "robust verification" for the MMS approach.

For further discussion on the possibility of a useful theorem related to MMS, see (Roache 2002, 2009). For the extensive discussion of V&V issues related specifically to modeling of nuclear waste disposal, see Roache (1998a Appendix C). For extensive discussion on semantics of V&V in computational physics specifically related to Popper's philosophy, see Roache (2012) and Chap. 27 by Roache in this volume. For the discussion of some current issues in V&V, including climate modeling, see Roache (2016).

12.6.7 Mere Mathematics

Rider (2018) noted that "the truism that verification is a purely mathematical exercise often works against verification. This is often used as an excuse to diminish its priority in code development. For codes used for science and engineering saying that it's just math can be used to say it's not important. This is unfortunate, but needs to be acknowledged and dealt with head-on."

We might expect that model developers would want some assurance that the code actually solved their model correctly, maybe even before they compared results to validation experiments! Furthermore, many validation exercises do not compare point values of all solution variables but only solution functionals, e.g., total heat flux. Especially in such comparisons, it is possible to achieve satisfactory agreement at particular experimental set points (i.e., values of experimental parameters) even though the code may have nontrivial errors.

In such situations, model developers might claim successful validation of their model M1 but in fact the code may contain an error E1. The actual "model" that is

"validated" is not M1 but some M2 = M1 + E1, where E1 is unknown to the developers. The result is a contradiction of a fundamental tenet of science: reproducibility. Other code developers who incorporate Model M1 correctly will not obtain the same results, for better or worse.

In recent history of fluid dynamics, it has been difficult to achieve the same results from different codes that ostensibly incorporate the same RANS turbulence model due to coding errors and to incomplete specification (documentation) of model details.

12.6.8 Irrelevance of Solution Realism to Code Verification

MMS generates solutions with no required concern for realism of the solution. Thus, acceptance requires that the judge recognize code verification as purely mathematical exercise. Physical realism and even realizability are irrelevant. Actually, there is no *requirement* that the MMS solution look unrealistic, and we can invent appealing solutions if necessary to satisfy managers, regulators, public stakeholders, etc. But it is worthwhile to understand that this "realism" is mere window dressing when we consider only the legitimacy of code verification per se. Solution realism is also risky in that it encourages a dangerous misconception, invites criticism and arguments about what constitutes "adequate realism" (surely a qualitative concept), and ostensibly justifies piecemeal and perpetual code verification exercises.

Furthermore, realistic solutions can actually be less desirable because often they only weakly exercise some terms, e.g., streamwise second derivatives in boundary layers. For the purpose of detecting ordered errors, it is best that the different solution terms in the governing equations be very roughly the same size. (An order of magnitude variation is not problematical.) As pointed out by Rider (2018), this is easier to control with wisely chosen unrealistic MMS solutions than with many classical solutions.

12.7 Reasons for Solution Realism in MMS

In spite of my claims above that MMS "solution realism" is irrelevant to legitimacy of code verification per se, it is also true that there are uses for realism, both inside and outside of code verification.

12.7.1 Realistic MMS in Code Verification of Glacial Ice Flow Modeling

Bueler et al. (2007) developed a realistic MMS solution to verify a code for solving glacial ice flows based on shallow (thin-film) ice approximations. Solution realism was important to gain acceptance at a time when the glaciology science community was skeptical of models and verifications. The 3D time-dependent model involves many difficult features: a free boundary, thermo-mechanical coupling between a highly nonlinear power law viscosity and the temperature distribution, and coupling between energy conservation and thin-layer mass conservation PDEs with integrals in the nonlinear PDE coefficients.

MMS was applied by starting with an exact solution to an isothermal ice model and then manufacturing a coupled exact solution from it (see Sect. 12.3.1). Solution realism aided interpretation of controversial temperature "spokes" in ice flows found by several investigators.

The paper contains highly detailed descriptions, unusual for an archive journal not devoted to V&V, of the implementation, advantages and disadvantages of the MMS procedures. The authors state that the glaciology community could substantially replace intercomparison of codes with true code verification using legitimate MMS exact solutions.

A subsequent model was described in Bueler and Brown (2009). Development of the University of Alaska—Fairbanks Parallel Ice Sheet Model continues and the open-source PISM code (www.pism-docs.org) has been widely used in climate modeling. The MMS verification procedure is built into the system and is used in daily regression code verifications.

12.7.2 Realistic MMS in Solution Verifications and Turbulence Models

MMS is applicable to code verification but not to solution verification per se. However, in devising *methods* for solution verification, MMS can play an important role in tuning empirical parameters for the classic Grid Convergence Index (GCI) method and variations (Roache 1993, 1998a, b, 2009). MMS has also contributed to estimation of errors due to incomplete iteration and outflow boundary conditions, and to evaluating solution adaptive grid generation methods (Eça and Hoekstra 2007b, 2009; Pelletier et al. 2004, Roache 2009). Many benchmark-quality solutions are required to achieve statistical significance, and each solution requires expensive brute-force discretization convergence computations. MMS solutions, if they are realistic, can be used to economically obviate the need for such expensive fine-grid solutions.

This approach is especially effective for evaluating turbulence models. However, it is far from a straightforward application of MMS. RANS turbulence models are especially difficult due to discontinuous switches, min/max functions, and strongly

nonlinear terms. As noted by Eça et al. (2007a, b), in a typical RANS model, there are no linear terms!

Eça and Hoekstra (2007a) and Eça et al. (2007a, b) used realistic MMS to study wall-bounded turbulence in 2D separated flows using both 1 and 2 equation RANS models; not surprisingly, they showed that the RANS models were inadequate in the near-wall region. Eça et al. (2007a, b) published detailed MMS solutions for several RANS models in conjunction with the Lisbon V&V Workshops (Eça et al. 2009). The benchmark realistic solutions and MMS source codes for six RANS models are available at the University of Lisbon website (Eça 2006). See Roache (2009) for additional references on the Lisbon Workshops.

Pelletier et al. (2004) used realistic MMS to tackle two of these difficult problems at once, turbulence models and solution adaptive FEM mesh generation, in the simulation of impinging round jets. They used the k-ε turbulence model and manufactured solutions for turbulent kinetic energy, eddy viscosity, and velocity.

12.7.3 Realistic MMS in Singularity Studies

Sinclair et al. (2006) independently developed realistic MMS (termed Tuned Test Problems) to evaluate methods for treatment of singularities during grid convergence studies. The techniques developed automatically detect and distinguish between cases of TTP-specified power singularities, logarithmic singularities, or simply grids not yet in the asymptotic range. For a summary, see Roache (2009, Sect. 5.4.10.1).

12.7.4 Other Uses and Generation Methods for Realistic MMS

Oberkampf and Roy (2010, Sect. 6.4, p. 235) note other cases in which physically realistic exact MMS solutions are desired: assessing sensitivity of a numerical scheme to mesh quality, and evaluating the reliability of discretization error estimators, as well as judging the overall effectiveness of solution adaptation schemes (see above). They describe two main approaches to generating realistic MMS solutions: theory-based solutions (see Sects. 12.3.1 and 12.7.1), and the Method of Nearby Problems. (The latter does not produce a single global analytical solution and has not seen much use.)

12.8 Alternative Formulations and General References for MMS

The basic *inverse* concept of MMS is to complicate the original problem a little to manufacture an intended solution; source terms are most straightforward and universally applicable. Another approach to MMS developed by Knupp and Salari (2003) is applicable to variable coefficient problems, e.g., groundwater transport or heat conduction codes with variable properties. A solution is manufactured by solving inversely for the distribution of variable coefficients that produce it.

Doebling (2016) verified a Lagrangian hydrodynamics code using an old (Fickett and Rivard 1974) exact solution for detonation problems. It was not described as MMS, and does not use manufactured source terms. But by (p. 1) "judicious selection of the material specific heat ratio, the problem has an exact solution with linear characteristics."

Burg and Murali (2004, 2006) developed a "residual formulation of MMS". The manufactured exact solution sets the initial condition, and only one iteration is used to evaluate the residuals. The residuals contain information on p in the sense of a Taylor's series expansion. But this approach does not actually verify the observed accuracy of a code since no solution is produced. While somewhat helpful for identifying locations of coding errors, the approach is not convincing for robust code verification, in my opinion.

Other general expositions of MMS are given in Knupp and Salari (2003), Roy (2005), Pelletier and Roache (2006), Wang et al. (2009), Oberkampf and Roy (2010). Besides the library of MMS solutions for turbulence (Eça 2006) already cited, Malaya et al. (2013) have created a library of code verification solutions including MMS as well as analytical solutions.

12.9 Conclusion

The Method of Manufactured Solutions for code verification was often met early with skepticism, but is now widely accepted. MMS enables one to produce many exact analytical solutions for use as benchmarks in systematic discretization refinement tests, which tests are remarkably sensitive for code verification. The method is straightforward and, when applied to all option combinations in a code, can lead to robust code verification with a strong completion point. It eliminates the typical haphazard, piecemeal and never-ending approach of partial code verifications with various highly simplified traditional problems that still leave the user unconvinced. Although the method requires some up-front work to become adept at using Symbolic Manipulation software, once achieved, this training time can be amortized over verifications of many codes. Producing a reliable and general MMS solution that exercises all the relevant terms typically involves more work than a single traditional solution, but if one considers the suite of traditional problems often used, then the

amount of work can be less using MMS. The MMS approach is especially useful and reduces the book-keeping and total workload when used for regression verification of code modifications affecting option combinations.

Acknowledgements I gratefully acknowledge help from C. Beisbart, L. Eça, P. Moin, W. L. Oberkampf, C. J. Roy, N. Saam, L. Shunn. and especially W. J. Rider.

References

ASME. (2006). *ASME V&V 10-2006. Guide for verification and validation in computational solid dynamics.*

ASME. (2009). *ASME V&V 20-2009. Standard for verification and validation in computational fluid dynamics and heat transfer*.

Blackwell, B., Dowding, K., & Modest, M. (2009). Cylindrical geometry verification problem for enclosure radiation. *Journal of Thermophysics and Heat Transfer, 23,* 711–715. https://doi.org/10.2514/1.39861.

Bond, R. B., Ober, C. C., Knupp, P. M., & Bova, S. W. (2007). Manufactured solution for computation fluid dynamics boundary condition verification. *AIAA Journal, 45*(9), 2224–2236.

Brady, P. T., Herrmann, M., & Lopez, J. M. (2012). Code verification for finite volume multiphase scalar equations using the method of manufactured solutions. *Journal of Computational Physics, 231,* 2924–2944.

Burg, C. O. E., & Murali, V. K. (2004). *Efficient code verification using the residual formulation of the method of manufactured solutions. AIAA Paper 2004-2628*, 34th AIAA Fluid Dynamics Conference, Portland, Oregon, June, 2004.

Burg, C. O. E., & Murali, V. K. (2006). The residual formulation of the method of manufactured solutions for computationally efficient code verification. *International Journal of Computational Fluid Dynamics, 20*(7), 2006.

Bueler, E., Brown, J., & Lingle, C. (2007). Exact solutions to the thermomechanically coupled shallow-ice approximation: effective tools for verification. *Journal of Glaciology, 53*(182), 499–516.

Bueler, E., Brown, J. (2009). Shallow shelf approximation as a "sliding law" in a thermomechanically coupled ice sheet model. *Journal of Geophysical Research*, *114*, F03008. https://doi.org/10.1029/2008jf001179.

Choudhary, A., Roy, C. J., Dietiker, J.-F., Shahnam, M. & Garg, R. (2014). Code verification for multiphase flows using the method of manufactured solutions, FEDSM2014-21608. In *Proceedings of the ASME 2014 4th Joint US-European Fluids Engineering Division Summer Meeting (FEDSM)*. Chicago, IL, August 3–7, 2014.16 M3.

Choudhary, A., Roy, C. J., Luke, E. A., & Veluri, S. P. (2016). Code verification of boundary conditions for compressible and incompressible computational fluid dynamics codes. *Computers & Fluids, 126,* 153–169.

Doebling, S. W. (2016). The escape of high explosive products: an exact-solution problem for verification of hydrodynamics codes. *Journal of Verification, Validation and Uncertainty Quantification, 1,* 041001–1–041001–13.

Eça, L. (2006). Workshop Website. http://maretec.ist.utl.pt/~maretec.daemon/html_files/CFD_workshops/Workshop_2006.htm.

Eça, L., & Hoekstra, M. (2007a). Evaluation of numerical error estimation based on grid refinement studies with the method of manufactured solutions. Report D72-42, MARIN, May 2007.

Eça, L., & Hoekstra, M. (2007b). *Code verification of unsteady flow solvers with the method of manufactured solutions*. Paper No. ISOPE-2007-565, International Society of Offshore and Polar Engineers.

Eça, L., & Hoekstra, M. (2009). Evaluation of numerical error estimation based on grid refinement studies with the method of manufactured solutions. *Computers and Fluids*, https://doi.org/10.1016/j.compfluid.2009.01.003.

Eça, L., Hoekstra, M., Hay, A., & Pelletier, D. (2007a). A manufactured solution for a two-dimensional steady wall-bounded incompressible turbulent flow. *International Journal of Computational Fluid Dynamics, 21,* 175–188.

Eça, L., Hoekstra, M., Hay, A., & Pelletier, D. (2007b). On the construction of manufactured solutions for one and two-equation eddy-viscosity models. *International Journal for Numerical Methods in Fluids, 54,* 119–154.

Eça, L., Hoekstra, M., Roache, P. J., & Coleman, H. (2009). *Code verification, solution verification and validation: An overview of the 3rd Lisbon Workshop*. AIAA Paper No. 2009-3647, 19th AIAA Computational Fluid Dynamics, San Antonio, Texas, June 2009.

Fickett, W., & Rivard, C. (1974). *Test problems for hydrocodes*. Los Alamos, New Mexico:Los Alamos Scientific Laboratory, Report No. LA-5479.

Grier, B., Alyanak, E., White, M., Camberos, J., & Figliola, R. (2014). Numerical integration techniques for discontinuous manufactured solutions. *Journal of Computational Physics, 278,* 193–203.

Grier, B., & Figliola, R. (2015). Discontinuous solutions using the method of manufactured solutions on finite volume solvers. *AIAA Journal, 53,* 2369–2378.

Grismer, M. J., & Powers, J. M. (1996). Numerical predictions of oblique detonation stability boundaries. *Shock Waves, 6,* 147–156.

IEEE. (1991). *IEEE standard glossary of software engineering terminology*, IEEE Std 610.12-1990, New York, IEEE.

Knupp, P., & Salari, K. (2003). *Verification of computer codes in computational science and engineering*. Boca Raon, FL: CRC Press.

Malaya, N., Estacio-Hiroms, K. C., Stogner, R. H., Schulz, K. W., Bauman, P. T., & Carey, G. F. (2013). MASA: A library for verification using manufactured and analytical solutions. *Engineering with Computers, 29,* 487–496.

Murali, V., Burg, C. O. E. (2002). *Verification of 2D navier-stokes codes by the method of manufactured solutions*. AIAA Paper 2002-3109, 32nd AIAA Fluid Dynamics Conference, St. Louis, June, 2002.

Oberkampf, W. L., & Trucano, T. G. (2002). Verification and validation in computational fluid dynamics. *AIAA Progress in Aerospace Sciences*.

Oberkampf, W. L., & Roy, C. J. (2010). *Verification and Validation in Scientific Computing*. Cambridge, UK: Cambridge University Press.

Pautz, S. D. (2001). Verification of transport codes by the method of manufactured solutions: The ATTILA experience. In *Proceedings of the ANS International Meeting on Mathematical Methods for Nuclear Applications, M&C 2001*. Salt Lake City, Utah, Sept 2001.

Pelletier, D., & Roache, P. J. (2006). Verification and validation of computational heat transfer. In W. J. Minkowycz, E. M. Sparrow, & J. Y. Murthy (Eds.), *Handbook of Numerical Heat Transfer* (2nd ed.). New York:Wiley.

Pelletier, D., Turgeon, E., & Tremblay, D. (2004). Verification and validation of impinging round jet simulations using an adaptive FEM. *International Journal for Numerical Methods in Fluids, 44,* 737–763.

Polya, G. (1957). *How to solve it, a new aspect of mathematical method*. Princeton, NJ: Princeton University Press.

Rider, W. J. (2018). Personal communication 5/5/2018.

Roache, P. J. (1993). A method for uniform reporting of grid refinement studies, ASME FED-Vol. 158. In I. Celik, C. J. Chen, P. J. Roache, & G. Scheurer (Eds.), *Quantification of uncertainty in computational fluid dynamics*. ASME Fluids Engineering Division Summer Meeting, Washington, DC, 20–24 June 1993, pp. 109–120.

Roache, P. J. (1998a). *Verification and validation in computational science and engineering*. Albuquerque, NM: Hermosa Publishers.

Roache, P. J. (1998b). *Fundamentals of computational fluid dynamics*. Albuquerque, NM: Hermosa Publishers.
Roache, P. J. (2002). Code verification by the method of manufactured solutions. *ASME Journal of Fluids Engineering, 114*(1), 4–10.
Roache, P. J. (2004). Building PDE codes to be verifiable and validatable. *Computing in science and engineering*. Special Issue on Verification and Validation, September/October 2004, 30–38.
Roache, P. J. (2009). *Fundamentals of verification and validation*, Hermosa Publishers, Albuquerque, NM, Ch. 3 and Appendix C.
Roache, P. J. (2012). *A defense of computational physics*. Albuquerque, NM: Hermosa Publishers.
Roache, P. J. (2016). Verification and validation in fluids engineering: some current issues. *ASME Journal of Fluids Engineering*. FE-16-1206. https://doi.org/10.1115/1.4033979.
Roache, P. J., & Steinberg, S. (1984). Symbolic manipulation and computational fluid dynamics. *AIAA Journal, 22*(10), 1390–1394.
Roy, C. J. (2001). *Grid convergence error analysis for mixed-order numerical schemes*. AIAA Paper 2001–2606, June 2001 (Anaheim).
Roy, C. J. (2005). Review of code and solution verification procedures for computational simulation. *Journal of Computational Physics, 205*(1), 131–136.
Roy, C. J. (2015). Code verification: past, present and future, keynote lecture. In *ASME V&V Symposium*, La Vegas, NV, 13 May 2015.
Roy, C. J., McWherter-Payne, M. A., & Oberkampf, W. L. (2000). Verification and validation for laminar hypersonic flowfields, AIAA 2000-2550, June 2000 (Denver).
Salari, K. & Knupp, P. (2000). Code verification by the method of manufactured solutions, SAND2000-1444, Sandia National Laboratories, Albuquerque, NM 87185, June 2000.
Shunn, L., Ham, F., & Moin, P. (2012a). Verification of variable-density flow solvers using manufactured solutions. *Journal of Computational Physics, 231*(9), 3801–3827.
Shunn, L., Ham, F., & Moin, P. (2012b). Verification of variable-density flow solvers using manufactured solutions. *Journal of Computational Physics, 231*(9), 3801–3827.
Sinclair, G. B., Beisheim, J. R., & Sezer, S. (2006). Practical convergence-divergence checks for stresses from FEA. In *Proceedings of the 2006 international ANSYS users conference and exposition*, 2–4 May 2006, Pittsburgh, PA.
Steinberg, S., & Roache, P. J. (1985). Symbolic manipulation and computational fluid dynamics. *Journal of Computational Physics, 57*(2), 251–284.
Steinberg, S., & Roache, P. J. (1986a). Variational grid generation. *Numerical Methods for Partial Differential Equations, 2,* 71–96.
Steinberg, S., & Roache, P. J. (1986b). Grid generation: A variational and symbolic-computation approach. In *Proceedings numerical grid generation in fluid dynamics conference*, July 1986, Landshut, W. Germany.
Steinberg, S., & Roache, P. J. (1992). Variational curve and surface grid generation. *Journal of Computational Physics, 100*(1), 163–178.
Wang, S. S. Y., Jia, Y., Roache, P. J., Smith, P. E., & Schmalz, R. A. Jr., (Eds.). (2009). *Verification and validation of 3D free-surface flow model*. ASCE/EWRI Task Committee.
Woods, C. N., & Starkey, R. P. (2015). Verification of fluid-dynamic codes in the presence of shocks and other discontinuities. *Journal of Computational Physics, 294,* 312–328.