Journal of Computational Physics 243 (2013) 46–57

Contents lists available at SciVerse ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

External verification analysis: A code-independent verification technique for unsteady PDE codes

Daniel Ingraham [*], Ray Hixon

University of Toledo, MIME Department, 2801 W. Bancroft St., MS 312, Toledo, OH 43606, USA

ARTICLE INFO

Article history:
Received 7 May 2012
Received in revised form 7 December 2012
Accepted 16 February 2013
Available online 16 March 2013

Keywords:
Code verification
Computational Fluid Dynamics
Computational Aeroacoustics
Time-marching schemes
Spatial differencing schemes

ABSTRACT

The External Verification Analysis (EVA) code verification technique is developed and presented. An extension of the Method of Manufactured Solutions (MMS), EVA provides a reference solution that can be used to verify both the spatial and temporal discretization schemes of an unsteady PDE solver, without the addition of arbitrary source terms. This feature of EVA allows the code to truly be treated as a "black box" during verification, simplifying the verification procedure. Currently, however, the EVA technique solves the interior equations of a PDE code only – thus the implementation of boundary conditions in the PDE code is not verified. The mathematical foundation of EVA is explained in detail, and the process used to verify a PDE code is discussed. The EVA technique is then demonstrated on a high-order, nonlinear Computational Aeroacoustics code.

1. Introduction

The application of the computer to the solution of the partial differential equations governing physical phenomena has produced a revolution in scientific research and engineering practice. With each passing year PDE codes become more powerful, allowing the simulation of increasingly complex and realistic physical processes. For instance, in the 1950s, a new algorithm for solving Poisson's equation was publication-worthy or could be the foundation of a doctoral dissertation [1], while today calculations involving the various forms of the Navier–Stokes equations are routinely performed by students using one of the many general-purpose Computational Fluid Dynamics codes [2,3] available. This impressive progress has been in part due to expanding computing power (the oft-mentioned Moore's Law [4]); but a review of, for example, the CFD literature shows that improved algorithms are also responsible [5–8]. The combination of the rise of more sophisticated numerical techniques and the astounding pace of computer hardware development has brought better predictive capability at the price of increased complexity. Ensuring that these PDE codes are free of coding mistakes (or bugs) is thus very important, and is the subject of the growing research field of Verification and Validation (V&V).

Verification was compactly defined by Boehm [9] as "solving the equations right" — this is in contrast to Validation, which confirms that a code "solves the right equations". More specifically, *code* verification is defined as rigorously demonstrating that a code is solving its governing equations to the purported order-of-accuracy [10,11]. This is done through grid convergence/time step studies, where one varies an appropriate measure of discretization (a term found in [11] for grid spacing or time step size) and monitors the error for each code run. If the error approaches zero at the expected rate, the code is said to be verified, at least for the set of options or schemes used. This process has been shown to be extremely sensitive to coding mistakes [12] — once a code has been verified, one has very strong evidence that the implementation of the numerical schemes is free of errors.

* Corresponding author. Tel.: +1 419 357 6248.
E-mail address: dingraha@eng.utoledo.edu (D. Ingraham).

0021-9991/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jcp.2013.02.032

In order to perform code verification, one must calculate the error of multiple code runs using some reference solution. In the past, a researcher's only option was to find and use an exact solution to the governing equations (or perhaps compare with another code, which has its own problems [13]). Inevitably (else there would be no reason for the code) the exact solutions available were for simplified geometry and coordinate systems, specific initial and boundary conditions, or were difficult to evaluate practically [10]. Many exact solutions only exercise a limited number of terms in the governing equation, perhaps allowing coding mistakes to go undetected.

Steinburg and Roache [14] devised a clever approach to overcoming this lack of exact solutions called the Method of Manufactured Solutions (MMS). Instead of attempting to find a solution that satisfies the governing equation and associated boundary conditions exactly, they assumed (or manufactured) a solution that does *not* satisfy the governing equations. The assumed solution is then inserted into the governing equations and the resulting residual expression is added to the original governing equations as a source term. The modified governing equation now has the assumed (or "manufactured") solution as an *exact* solution. The code to be verified must then be modified to include the new source term of the manufactured solution, and the grid or time step studies can be performed using the manufactured solution as an exact solution. A clear introduction to the MMS technique and details of its application can be found in [15].

Since its development, the MMS technique has become a very popular code verification method. For example, in the realm of Computational Fluid Dynamics, MMS has been used to verify the steady [16] and unsteady Reynolds-Averaged Navier–Stokes equations [17], and CFD boundary conditions [18–20], free-surface flows [21], codes supporting fluid-structure interaction [22], and various commercial codes [23]. MMS is not limited to CFD codes: it has been used with transport codes [24] and aero-optics codes [25], and was originally applied to both a Poisson equation solver and grid generation code [14].

The Method of Manufactured solutions is not without its disadvantages, however. Proponents of the method point out that the algebra required to generate the source terms from the "manufacturing" process may be considerable (see Roy [26], in addition to Roache [11] and Salari and Knupp [12]); however, the popularity of computer algebra codes like Mathematica, Maple, or MatLAB have made this less of a problem. The main disadvantage, however, is that a code must be able to support a *distributed* (and perhaps unsteady) source term if MMS is to be used, and (unless one is clever with the chosen manufactured solution), arbitrary initial and boundary conditions [26]. If the code to be verified already has this capability, then the application of MMS should be straightforward — if it does not, then the code must be modified in some way. In some situations this may be undesirable: for instance, if the user is unfamiliar with the programming language the code uses, or just the code itself, it may be difficult to add the necessary capability to the PDE solver with a reasonable certainty that new coding mistakes have not been introduced. In the case of proprietary codes the source code may not be available at all, making the use of MMS very difficult if the code does not meet the procedure's requirements.

An ideal code verification method would allow one to overcome the lack of exact solutions available for the more "complex" PDEs without requiring modification of the source of the code one wishes to verify. Hixon and Li have recently developed a extension of MMS that meets these requirements called External Verification Analysis [27]. This new technique provides an alternative method of obtaining a reference solution that can be used with grid or time step studies to confirm the order-of-accuracy of the numerical algorithms used in a "marching" (e.g. unsteady) PDE code. The EVA tool solves the actual governing equations of interest, not a modified version as in MMS, and thus does not require the addition of arbitrary source terms to the code. The EVA tool is therefore completely *external* to the code — one can truly treat the PDE code as a "black box." In its current formulation, the EVA technique works with the interior governing equations of the PDE code only - it essentially evolves the solution in time on an infinite domain. Because of this fact, the EVA technique, as currently constructed, can only be used to verify the interior solver of a PDE code - boundary conditions are not verified. In this work, the EVA technique is presented, including results of its application on a high-order, nonlinear Computational Aeroacoustics (CAA) code.

2. The External Verification Analysis (EVA) process

The External Verification Analysis process can be used to verify a PDE code much like MMS — the only essential difference is how the "reference" solution for the grid/time step studies is obtained. Briefly, the steps of the process are:

1. Choose the test case parameters.
2. Use the EVA solution technique to obtain a "reference" solution.
3. Run the PDE code with the chosen parameters (grid, initial condition, etc.) with a range of "discretization measures" (grid spacing or time step).
4. Calculate the error between the results from the PDE code and the EVA tool.
5. Calculate the observed order of accuracy, and compare to the formal order of accuracy of the PDE code's numerical scheme.

Each of these steps will be described in detail in this section. But because many of the steps are influenced by the way the EVA tool obtains a reference solution, the EVA solution technique will be described first.