INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2004; **44**:599–620 (DOI: 10.1002/fld.660)

Verification of Euler/Navier–Stokes codes using the method of manufactured solutions

C. J. Roy[1,*,†], C. C. Nelson[2], T. M. Smith[3] and C. C. Ober[3]

[1]*Aerospace Engineering Department, Auburn University, AL 36849-5338, U.S.A.*
[2]*Arnold Engineering Development Center,[‡] 740 Fourth Street, Arnold AFB, TN 37389-6001, U.S.A.*
[3]*Sandia National Laboratories,[§] MS 0316, P.O. Box 5800, Albuquerque, NM 87185-0316, U.S.A.*

SUMMARY

The method of manufactured solutions is used to verify the order of accuracy of two finite-volume Euler and Navier–Stokes codes. The Premo code employs a node-centred approach using unstructured meshes, while the Wind code employs a similar scheme on structured meshes. Both codes use Roe's upwind method with MUSCL extrapolation for the convective terms and central differences for the diffusion terms, thus yielding a numerical scheme that is formally second-order accurate. The method of manufactured solutions is employed to generate exact solutions to the governing Euler and Navier–Stokes equations in two dimensions along with additional source terms. These exact solutions are then used to accurately evaluate the discretization error in the numerical solutions. Through global discretization error analyses, the spatial order of accuracy is observed to be second order for both codes, thus giving a high degree of confidence that the two codes are free from coding mistakes in the options exercised. Examples of coding mistakes discovered using the method are also given. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: Euler/Navier–Stokes equations; exact solution; code verification; manufactured solution; benchmarking; order of accuracy

1. INTRODUCTION

Modelling and simulation (M&S) has enormous potential to impact the design, analysis, and optimization of engineering systems. Here M&S is viewed as the numerical solution to any set of partial differential equations that govern continuum mechanics or energy transport (e.g.

*Correspondence to: C. J. Roy, Aerospace Engineering Department, Auburn University, 211 Aerospace Engineering Building, Auburn, AL 36849-5338, U.S.A.
†E-mail: cjroy@eng.auburn.edu
‡The research reported herein was performed by the Arnold Engineering Development Center (AEDC), Air Force Materiel Command. Work and analysis for this research were performed by personnel of Sandia National Laboratories and Jacobs Sverdrup AEDC Group, technical services contractor for AEDC. Further reproduction is authorized to satisfy needs of the U.S. Government.
§Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Received 24 March 2003
Revised 15 September 2003

structural dynamics, heat conduction, electrostatics, fluid dynamics). In order for M&S to fully achieve its potential, the engineering community must gain increased confidence that it can provide accurate predictions. Although the specific examples presented herein are for Computational Fluid Dynamics (CFD), the general concepts apply to any M&S code.

The sources of error in M&S can be categorized into two distinct areas [1, 2], physical modelling errors (validation related) and mathematical errors (verification related). The physical modelling errors arise due to shortcomings in the chosen model or when a model is applied outside of its intended range. An example of the latter is a turbulence model which provides surface heating results for an attached boundary layer flow within 10% accuracy. When this model is applied outside the range where it was calibrated, say for shock-induced separation, then the model may only give 50% accuracy on heating rates.

Mathematical or verification-related errors can arise from a number of sources including insufficient mesh resolution, improper time step, incomplete iterative convergence, round-off and coding mistakes (i.e. coding bugs). The presence of coding mistakes is an often overlooked source of error in M&S. Code developers often rely on expert judgement to determine when a code is producing the correct results. As M&S codes become more complex with numerous modelling options and multiphysics coupling, the reliance on expert judgement can be problematic. This paper will discuss a rigorous method for finding and eliminating coding mistakes known as the method of manufactured solutions [1, 3].

Verification is defined as ensuring that a model implementation matches the developer's conception [2]. Verification can be broken down into two distinct categories: code verification and solution verification. Code verification is a process performed to provide a high degree of certainty that a code is free from coding mistakes (i.e. coding bugs); however, a formal proof that a piece of software is 'bug-free' is probably not forthcoming. If performed rigorously, the code verification process needs to be performed only once for each independent portion of the code, assuming no subsequent changes to the code are made.

There are four different ways to verify a code: the method of exact solutions, the method of manufactured solutions, comparison to benchmark numerical solutions and code-to-code comparisons. The latter two are really confidence-building exercises, and should not take the place of rigorous code verification. In the method of exact solutions, numerical solutions are compared to exact solutions, often with simplifications to the equations and/or the boundary conditions. In the method of manufactured solutions, an analytical solution is chosen *a priori* and the governing equations are modified by the addition of analytical source terms.

The use of manufactured solutions and grid convergence studies for the purposes of code verification was first proposed by Roache and Steinberg [4]. They employed the symbolic manipulation software Macsyma to verify a code for generating three-dimensional transformations for elliptic partial differential equations. These concepts were later extended by Roache *et al.* [5]. The term "manufactured solution" was coined by Oberkampf and Blottner [6] and refers to the fact that the method generates (or manufactures) a related set of governing equations to a chosen analytic solution. An extensive discussion of manufactured solutions for code verification was presented by Salari and Knupp [7], and includes both details of the method as well as application to a variety of partial differential equation sets. This report was later refined and published in book form by Knupp and Salari [3]. A recent review/tutorial was also given by Roache [8].

The first CFD code to be verified in the current work is the Premo code, which is being developed as part of the Department of Energy's Accelerated Strategic Computing Initiative

(ASCI) to meet the needs of the Stockpile Stewardship Program. The Premo code is one of a number of mechanics and energy transport codes that serves as a module to the SIERRA multi-mechanics framework [9]. The SIERRA framework provides services for I/O, domain decomposition, massively parallel processing, mesh adaptivity, load balancing, code coupling and interfaces to a host of linear and non-linear solvers.

The second CFD code to be verified is version 6.0 of the Wind code [10] (projected to be released in 2004), the primary component of the National Project for Applications-oriented Research in CFD (NPARC) Alliance Flow Simulation System. Wind is derived from the NASTD code, which was donated by the Boeing Company to NPARC in 1997. Wind is developed and maintained jointly by the Arnold Engineering Development Center, NASA Glenn Research Center and the Boeing Company under the auspices of the NPARC Alliance.

2. NUMERICAL FORMULATION

2.1. Euler equations

The two-dimensional Euler equations in conservation form are

$$\frac{\partial(\rho)}{\partial t} + \frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} = f_m \tag{1}$$

$$\frac{\partial(\rho u)}{\partial t} + \frac{\partial(\rho u^2 + p)}{\partial x} + \frac{\partial(\rho uv)}{\partial y} = f_x$$

$$\frac{\partial(\rho v)}{\partial t} + \frac{\partial(\rho vu)}{\partial x} + \frac{\partial(\rho v^2 + p)}{\partial y} = f_y \tag{2}$$

$$\frac{\partial(\rho e_t)}{\partial t} + \frac{\partial(\rho u e_t + pu)}{\partial x} + \frac{\partial(\rho v e_t + pv)}{\partial y} = f_e \tag{3}$$

where ρ is the mass density, u and v the Cartesian velocity components, p the static pressure, and e_t the total energy (internal plus kinetic). The first term is the unsteady term, the next two terms are the convective terms in the x and y directions, respectively, and a general source term is included on the right-hand side. For a calorically perfect gas, the Euler equations are closed with two auxiliary relations for energy

$$e = \frac{1}{\gamma - 1} RT \tag{4}$$

$$e_t = e + \frac{u^2 + v^2}{2} \tag{5}$$

and with the ideal gas equation of state

$$p = \rho RT \tag{6}$$

where T is the temperature. For the solutions presented herein, the ratio of specific heats is $\gamma = 1.4$ and the specific gas constant is $R = 287.0\ \mathrm{N\,m/(kg\,K)}$.

2.2. Navier–Stokes equations

For viscous flows, the mass conservation equation given in Equation (1) is still valid; however, the inviscid momentum equations are replaced by the Navier–Stokes equations, which may be written in two-dimensional form as

$$\begin{aligned} \frac{\partial(\rho u)}{\partial t} + \frac{\partial(\rho u^2 + p - \tau_{xx})}{\partial x} + \frac{\partial(\rho uv - \tau_{xy})}{\partial y} &= f_x \\ \frac{\partial(\rho v)}{\partial t} + \frac{\partial(\rho vu - \tau_{xy})}{\partial x} + \frac{\partial(\rho v^2 + p - \tau_{yy})}{\partial y} &= f_y \end{aligned} \tag{7}$$

Including the viscous effects, the energy conservation equation is now

$$\begin{aligned} &\frac{\partial(\rho e_\mathrm{t})}{\partial t} + \frac{\partial(\rho u e_\mathrm{t} + pu - u\tau_{xx} - v\tau_{xy} + q_x)}{\partial x} \\ &\quad + \frac{\partial(\rho v e_\mathrm{t} + pv - u\tau_{xy} - v\tau_{yy} + q_y)}{\partial y} = f_e \end{aligned} \tag{8}$$

For the two-dimensional Navier–Stokes equations, the shear stress tensor is

$$\begin{aligned} \tau_{xx} &= \frac{2}{3}\mu\left(2\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right) \\ \tau_{yy} &= \frac{2}{3}\mu\left(2\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x}\right) \\ \tau_{xy} &= \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \end{aligned} \tag{9}$$

and the heat flux vector is given by

$$\begin{aligned} q_x &= -k\frac{\partial T}{\partial x} \\ q_y &= -k\frac{\partial T}{\partial y} \end{aligned} \tag{10}$$

For the Navier–Stokes simulations presented herein, the absolute viscosity is chosen to be a large constant value ($\mu = 10\ \mathrm{N\,s/m^2}$) in order to obtain a balance between convection and diffusion. The thermal conductivity k is determined from the viscosity by specifying a constant

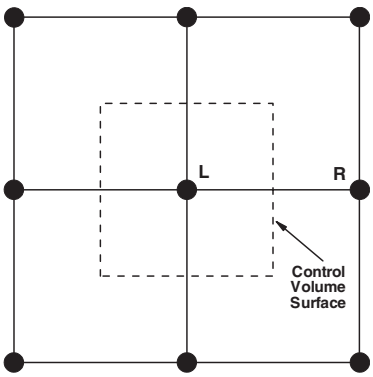

Figure 1. Edge-based control-volume discretization scheme (Premo code).

Prandtl number ($Pr = 1$):

$$k = \frac{\gamma R}{\gamma - 1} \frac{\mu}{Pr} \tag{11}$$

2.3. Discretization

The spatial discretization employed in the Premo code is a node-centred finite-volume formulation [11]. This discretization is implemented on unstructured meshes using an edge-based scheme which allows arbitrary element topologies, where an element is determined by connecting nearest-neighbour nodes. The surfaces of the control volume are found by connecting nodal edge mid-points and element centroids, i.e. the median dual mesh. The convective fluxes are evaluated with Roe's approximate Riemann solver [12]. Second-order spatial accuracy is achieved via MUSCL extrapolation [13] for the primitive variables to the control-volume surface. This extrapolation takes the form

$$\begin{aligned} \phi^- &= \phi_{\mathrm{L}} + \tfrac{1}{2}[\nabla\phi_{\mathrm{L}}] \bullet \Delta\vec{r} \\ \phi^+ &= \phi_{\mathrm{R}} - \tfrac{1}{2}[\nabla\phi_{\mathrm{R}}] \bullet \Delta\vec{r} \end{aligned} \tag{12}$$

where L and R refer to the nodes to the left and right of the control surface, the $-$ and $+$ refer to the left and right Riemann states, respectively, and $\Delta\vec{r}$ is distance vector

$$\Delta\vec{r} = (x_{\mathrm{R}} - x_{\mathrm{L}})i + (y_{\mathrm{R}} - y_{\mathrm{L}})j$$

The left and right states for the node-centred control volume are shown graphically in Figure 1. For the results presented herein, the gradient is evaluated using the least-squares gradient operator [14]. This gradient is also used in the evaluation of the viscous fluxes at the control-volume surface, resulting in a second-order discretization for the viscous terms. On uniform meshes, this scheme results in an upwind-biased Fromm's stencil for convection [15] and central difference for diffusion. For the simulations discussed in this paper, the governing equations are integrated in time to a steady state using a low-storage, four-stage Runge–Kutta

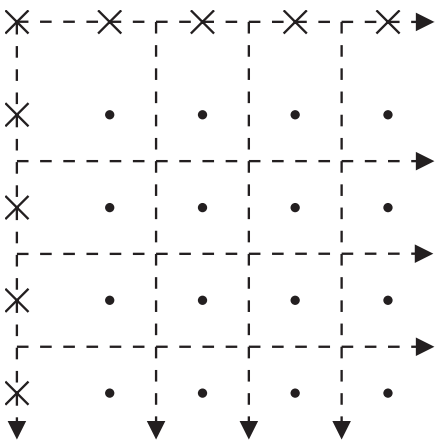

Figure 2. Grid nodes and control volumes on a uniform nodal mesh (Wind code).

method [16]. See Reference [11] for more details on the temporal and spatial discretization of the Premo code.

The Wind code also employs a finite-volume discretization in the solution of the governing equations. Interior mesh points are treated as cell centres and the corners that define a given control volume around them are computed by averaging the co-ordinates of neighbouring cell centres. With this information, cell volumes and face areas can be readily computed. On the boundaries, however, Wind uses a technique that was inherited from the NASTD code. Rather than solving half-cells at boundaries, the nearest interior cell is enlarged to reach all the way to the boundary. Therefore, if a uniform grid is input, this results in the so-called 'fat' cells at boundaries, as illustrated in Figure 2. These fat cells have two main drawbacks: first, they introduce a discontinuity in the grid spacing at the boundaries, and second, the cell centres of the fat cells are not actually in the centre of mass of the cell (even on a uniform grid). Both of these issues can introduce errors. In order to avoid problems with fat cells in the current work, a pre-processor was employed to read in a smooth, well-defined nodal grid, compute cell centres, and add a fringe of boundary points around the edges. This 'cell-centred' grid results in better behaved boundary cells as shown in Figure 3. For the cases reported here, this procedure yields cells which are truly uniform across the domain and cell centres that are exactly in the centre of mass of the cell. Thus, the Wind code behaves much like a cell-centred finite-volume code, although the internal algorithm remains node-centred. In order to compare the results from Wind and Premo in a consistent fashion, the solution at the original grid nodes was computed using an average of the values at the surrounding cell centres. Since this averaging procedure is a second-order process on uniform meshes, this should not affect the predicted order of the scheme, although the absolute magnitude of the error will change.

For the current computations, the Wind code, much like Premo, uses Roe's approximate Riemann solver for the evaluation of the convective fluxes. An approximate factorization implicit scheme is used to integrate the solution in time. For the Navier–Stokes results, the viscous fluxes are evaluated at cell faces using a central-difference scheme. This results in an overall scheme which, on uniform meshes, is formally second order in space. One major difference between Premo and Wind is that the latter cannot presently be run in double

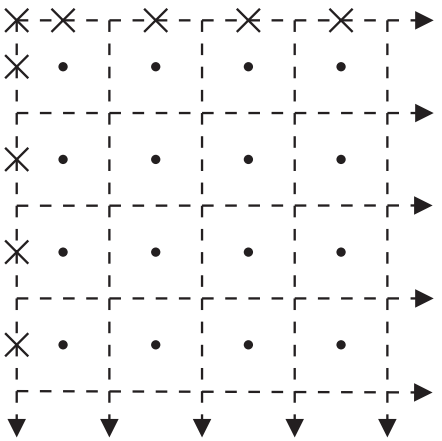

Figure 3. Grid nodes and control volumes on a nodal mesh modified to achieve uniform control volumes (Wind code).

precision (64 bit) mode. Thus, all Wind results shown here employ single precision (32 bit) computations. For more details regarding the numerics of Wind, see References [10, 17–19].

2.4. Boundary conditions

The Euler simulations are for supersonic flow and thus employ exact Dirichlet values for all primitive variables at the inflow boundaries based on the specified manufactured solution. For the Premo code, which employs a node-centred finite-volume discretization, the Dirichlet boundary condition is enforced by simply setting the nodal boundary value. For the outflow boundary nodes the weak form of the boundary condition is used where the nodal values are updated from the flux balance consistent with the Roe discretization. These nodal values are used for the entire control-volume surface along the outflow boundary. The Wind code uses a similar treatment at inflow boundaries, while first-order extrapolation of all the variables is used at the supersonic outflow boundaries.

For the Navier–Stokes simulations which employ subsonic flow, the viscosity was chosen to be a large value in order to ensure that the diffusive terms are on the same order as the convective terms (see Reference [20] for details). By setting the viscosity to a large value, the characteristic-based boundary conditions for inflow and outflow, which are based on one-dimensional inviscid theory, are no longer valid. Therefore, for the Navier–Stokes simulations, the exact Dirichlet values for all primitive variables are specified on both inflow and outflow boundaries.

3. CODE VERIFICATION

Code verification is a way to build confidence that there are no coding mistakes (or 'bugs') in a simulation code. While no rigorous proof of code verification currently exists (and none may be forthcoming), the procedures discussed in this section can provide a high degree of confidence that a code is mistake free. There is an ongoing debate as to whether code

verification is something that must be carried out only once for a given set of code options [1, 8], or whether code verification is a process by which code verification evidence is gathered to provide increased confidence that the code is 'bug' free [21]. The authors subscribe to the view that, if carried out properly, code verification need be performed only once for a given set of code options.

3.1. Method of exact solutions

In the method of exact solutions, specialized cases are identified where exact solutions exist to a given set of governing equations. For the Euler and Navier–Stokes sets of equations, there are only a limited number of exact solutions. Furthermore, these exact solutions may not exercise all of the terms in the governing equations. For example, in the flow between two infinite parallel plates, one moving relative to the other (Couette flow), the velocity profile is linear, hence the diffusion term, a second derivative of velocity, is identically zero and therefore would not be fully exercised.

3.2. Method of manufactured solutions

A more general approach to code verification is the method of manufactured solutions [1, 7]. With this approach, the mathematical form of the solution is chosen *a priori*. The differential operator for the governing equations is applied to this chosen analytical solution to generate analytical source terms. These source terms are implemented within the code, and the modified governing equations (including the source terms) are then discretized and solved numerically and compared to the exact solution.

There are a variety of acceptance criteria for code verification [3, 7]. In order of increasing rigor these criteria are:

- expert judgement,
- error quantification,
- consistency and
- order of accuracy.

The order of accuracy test is the most rigorous test and is therefore the recommended acceptance criteria. This test involves evaluating the numerical solution on a series of grids, and with various time steps for unsteady problems. The spatial (and/or temporal) discretization error is monitored to determine if the observed order of accuracy matches the formal order of accuracy, which can be determined by a truncation-error analysis of the discretized equations. The discretization error is defined as the difference between the solution to the discretized equations and the exact solution to the continuum partial differential equations. The discretization error will generally decrease as $1/r^p$, where r is the grid refinement factor (e.g. $r = \Delta x_{\text{coarse}}/\Delta x_{\text{fine}}$) and p is the order of accuracy. For example, if the numerical scheme is second order ($p = 2$) and the grid is doubled ($r = 2$), then the discretization error should decrease by a factor of four as the mesh is refined.

Although the form of the manufactured solution is somewhat arbitrary, it should be chosen to be smooth, infinitely differentiable and realizable (i.e. solutions should be avoided which have negative densities, pressures, temperatures, etc.). Solutions should also be chosen that are sufficiently general so as to exercise all terms in the governing equations. Adherence to

these guidelines will help ensure that the formal order of accuracy is attainable on reasonably coarse meshes. Symbolic manipulation tools such as Mathematica™ or Maple™ can be used to apply the differential operator to the solution and generate source terms. These tools can then be used to generate the FORTRAN or C coding for the source term automatically. See References [1, 7] for more information on the method of manufactured solutions.

While the method of manufactured solutions can be used to rigorously test the spatial and temporal discretization of a code, as well as the associated boundary conditions, this method will not test the efficiency or robustness of a given numerical solution method. In addition, the method cannot be used to test the stability of a given algorithm. Furthermore, options that are not exercised by the given manufactured solution are not verified.

The six steps required for implementing the method of manufactured solutions are:

(*Step* 1) Choose the form of the governing equations.
(*Step* 2) Choose the form of the manufactured solution.
(*Step* 3) Apply the governing equations to the manufactured solution to generate analytical source terms.
(*Step* 4) Discretize the equations and solve on multiple mesh levels using analytical boundary conditions and source terms from the manufactured solution.
(*Step* 5) Evaluate the global discretization error in the numerical solutions.
(*Step* 6) Determine whether or not the observed order of accuracy matches the formal order of accuracy.

If the comparison in Step 6 is favourable, then the coding options exercised are verified. If the comparison is unfavourable, then one generally examines the local discretization error, uses this information to debug the code, then returns to Step 4. The manufactured solution itself can be used to help debug the code by selectively turning off certain solution terms (viscous terms, a given spatial variation, etc.).

3.3. Order of accuracy

The most rigorous test for code verification is the order of accuracy test. This test assesses whether or not the numerical method reproduces the formal order of accuracy in space and/or time. As previously discussed, the discretization error should drop as $1/r^p$, where in the current case the grid refinement factor is $r=2$ and the nominal order of accuracy is $p=2$; thus, the error should drop by a factor of four on each successively refined mesh level. In order to examine the behaviour of the global discretization error, the discrete L_2 and L_∞ norms of the discretization error $(\phi_{k,n} - \phi_{\text{exact},n})$ are used:

$$L_2 \text{ norm}_k = \left(\frac{\sum_{n=1}^{N} |\phi_{k,n} - \phi_{\text{exact},n}|^2}{N} \right)^{1/2}$$

$$L_\infty \text{ norm}_k = \max |\phi_{k,n} - \phi_{\text{exact},n}| \tag{13}$$

where k refers to the discrete mesh level and n varies over all interior mesh nodes N on the coarsest mesh. Since the L_∞ norm represents the maximum discretization error over the

entire domain, obtaining the formal order of accuracy in this norm is generally more difficult than the other error norms. When these global error norms do not reproduce the formal order of accuracy, it is often helpful to examine the behaviour of the local discretization error.

The observed order of accuracy can be calculated given two discrete mesh levels (k and $k+1$) and the exact solution by

$$p_k = \ln\left(\frac{L_{k+1}}{L_k}\right) \Big/ \ln(r) \tag{14}$$

where 'L' refers to one of the discrete error norms from Equation (13), $k+1$ refers to the coarser mesh level, and r is the grid refinement factor. Once the order of accuracy is verified (assuming it is greater than zero), it is clear that the code is consistent in the sense that the numerical solution approaches the continuum solution (which was chosen in the beginning) as Δx and Δy approach zero.

4. RESULTS

This section reports manufactured solution results for the two-dimensional Euler and Navier–Stokes equations. Since the manufactured solutions exist for all x and y, we are free to choose any sub-domain on which to solve the governing equations. For the cases presented herein, the numerical solutions are obtained on the domain

$$0 \leqslant x/L \leqslant 1$$
$$0 \leqslant y/L \leqslant 1$$

with $L = 1$ m. The mesh sizes examined are given in Table I, where h_k is the ratio of the element size at the kth level to the element size at the finest level:

$$h_k = \frac{\Delta x_k}{\Delta x_1} = \frac{\Delta y_k}{\Delta y_1}$$

Note that since only uniform Cartesian meshes are examined, the codes cannot be said to be verified for arbitrary meshes.

4.1. Euler equations

The method of manufactured solutions is first applied to the Euler equations given by Equations (1)–(3) along with the auxiliary relations given in Equations (4)–(6). These equations

Table I. Meshes for the manufactured solutions.

Mesh	Mesh nodes	Grid spacing, h
1	129×129	1
2	65×65	2
3	33×33	4
4	17×17	8
5	9×9	16

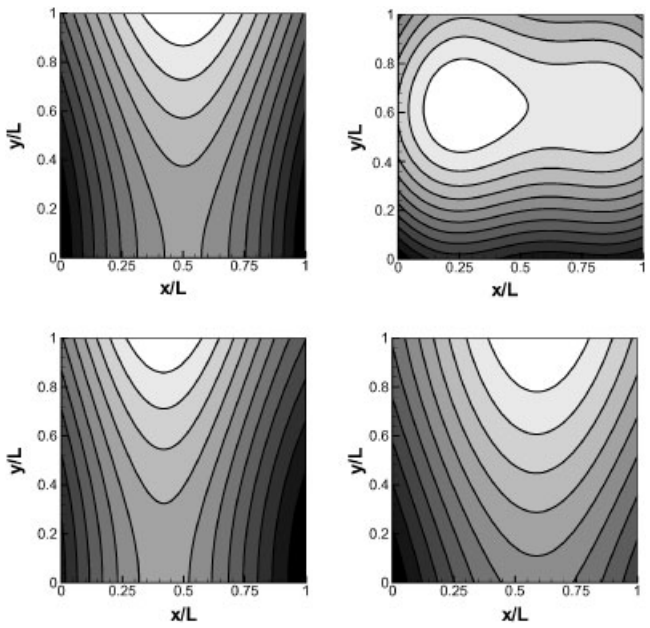

Figure 4. Supersonic Euler manufactured solution: ρ (top left), ρe_t (top right), ρu (bottom left) and ρv (bottom right).

govern the conservation of mass, momentum and energy for an inviscid (frictionless) fluid. With the governing equations specified, the next step is to choose the form of solution. The general form of the primitive solution variables is chosen as a function of sines and cosines

$$\phi(x, y) = \phi_0 + \phi_x f_s\left(\frac{a_{\phi x}\pi x}{L}\right) + \phi_y f_s\left(\frac{a_{\phi y}\pi y}{L}\right) + \phi_{xy}\, f_s\left(\frac{a_{\phi yx}\pi xy}{L^2}\right) \tag{15}$$

where $\phi = \rho, u, v$ or p for density, u-velocity, v-velocity or pressure, respectively, and $f_s(\cdot)$ denotes either the sine or cosine function. Note that in this case, ϕ_x, ϕ_y and ϕ_{xy} are constants (the subscripts do not denote differentiation). See Appendix A for the form of these primitive variables for the Euler and Navier–Stokes results presented herein. The chosen solutions are thus smoothly varying functions in space, while the temporal accuracy is not addressed in this study. The constants used in the manufactured solutions for the supersonic Euler case are given Table BI in of Appendix B. These solutions are presented graphically in Figure 4, and are chosen to be smooth to allow the formal order of accuracy to be achieved on relatively coarse meshes.

The governing equations (Equations (2)–(6)) were applied to the chosen solutions using the Mathematica™ symbolic manipulation software to generate FORTRAN code for the resulting source terms. The analytical source term for the Euler mass conservation equation is given in Appendix C. The source terms for each of the governing equations for this case are shown graphically in Figure 5.

The governing equations (Equations (2)–(6)) were then discretized and solved numerically, including the analytical source terms. For a given control volume, the source terms were

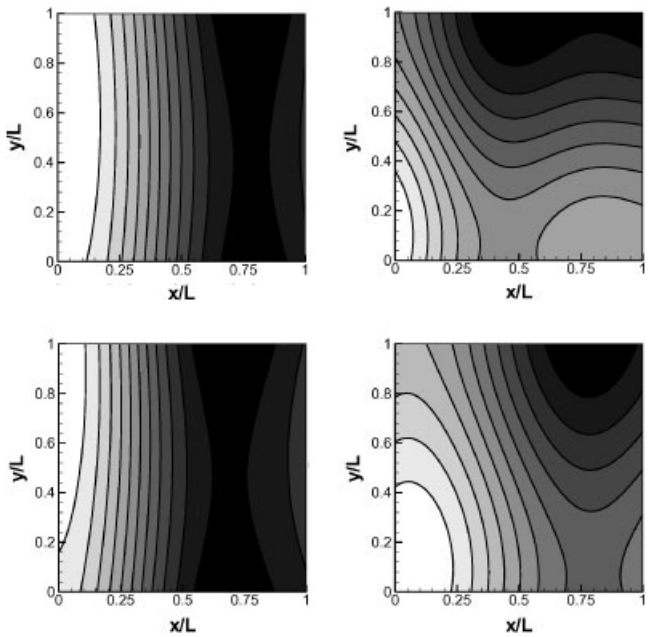

Figure 5. Generated source terms for the supersonic Euler case: mass (top left), energy (top right), x-momentum (bottom left), y-momentum (bottom right).

simply evaluated using the values at the control-volume centroid. The numerical solutions were then compared to the manufactured solution to determine the discretization error in the solutions. The formal order of accuracy of both the Premo and Wind codes is second order in space.

For the steady-state solutions examined, all solutions are iterated in pseudo-time to achieve iterative convergence. The most common method for judging iterative convergence is to examine the steady-state residuals of the governing equations. The steady-state residuals are defined by plugging the discrete solution for the current iteration into the discrete form of the steady-state equations (including the source terms). For non-linear equations, the relationship between the steady-state residuals and the iterative error in the dependent variables ($\rho, \rho u, \rho v$ and ρe_t) is difficult to determine and is related to the condition number of the discretized system. Both the steady-state residuals and the iterative errors will be examined.

The Premo solutions were marched to a steady-state solution using the four-stage Runge–Kutta temporal integration, while the Wind solutions employed the implicit approximate factorization scheme. All solutions presented herein were integrated in time until the L_2 norm of the steady-state residuals was converged to machine zero. The Premo code was run in double precision (64 bit) mode, and the steady-state residuals were reduced by at least 13 orders of magnitude from their initial levels. The Wind code was run in single precision (32 bit) mode, and the residuals were reduced by at least 6 orders of magnitude from their initial levels. Figure 6 shows the steady-state residual histories for the mass and energy equations on the 33×33 node mesh using both codes (lines).

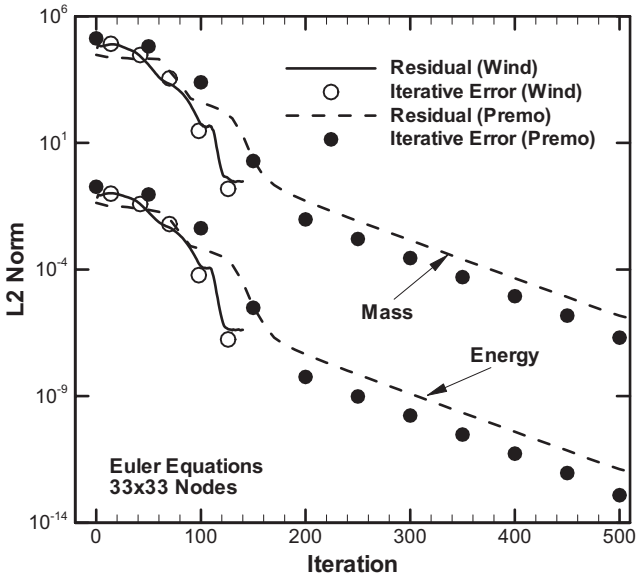

Figure 6. L_2 norms of the steady-state residuals (mass and energy equations) and the iterative error (ρ and ρe_t) for the Euler case.

Also shown in Figure 6 is the L_2 norm of the iterative error in the mass ρ and the energy ρe_t (symbols). The iterative error is defined as the difference between the computed solution at the current iteration and the computed solution at the final iteration. In all cases, the steady-state residuals are driven to machine zero by the final iteration. The steady-state residuals in Figure 6 are scaled so as to provide approximately the same order of magnitude as the iterative errors. It is clear from the figure that a drop in the residuals (the quantities most often examined in steady-state calculations) provides a corresponding drop in the iterative error. Similar iterative convergence was found for momentum (ρu and ρv). As a rule of thumb, the iterative error will not pollute the discretization error studies if the iterative errors are driven roughly two orders of magnitude smaller than the discretization error on all mesh levels.

The discretization error norms for this case are presented in Figure 7 for the mass density using both the Premo and Wind codes. The abscissa shows the measure of the grid spacing h on a log scale, with $h = 1$ being the finest mesh (129×129 nodes). Also shown on the plot are curves for the first- and second-order slopes for reference. Both the discrete L_2 and L_∞ norms of the discretization error (see Equation (13)) drop by a factor of four with each mesh refinement, thus matching the second-order slope and verifying that both codes are producing second-order accurate results. For both codes, the discretization error levels are at least two orders of magnitude larger than the iterative error levels from Figure 6 as desired.

The observed order of accuracy can be calculated given two discrete mesh levels (k and $k + 1$) and the exact solution by Equation (14). These order of accuracy results are shown graphically in Figure 8 which again show that the solutions are second-order accurate as the

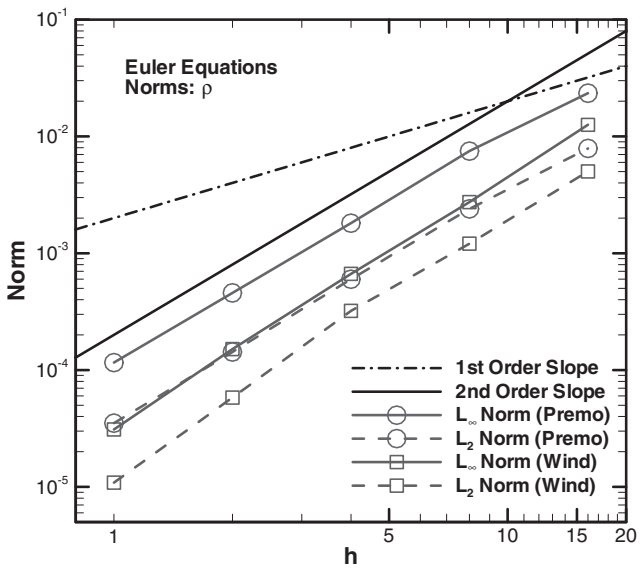

Figure 7. Behaviour of the density discretization error norms as the mesh is refined for the Euler case.

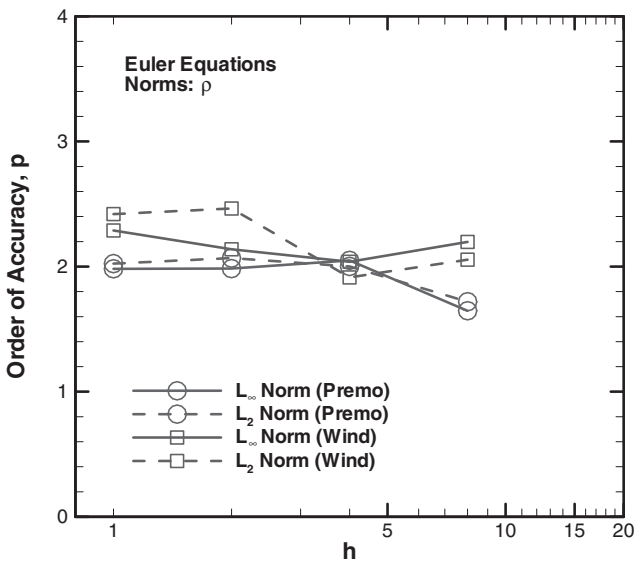

Figure 8. Observed order of accuracy of the density discretization error norms as the mesh is refined for the Euler case.

mesh is refined. In fact, the observed order of accuracy of the Wind code appears to be slightly higher than second order for this case. Although not shown, similar behaviour was found for the other conserved variables (ρu, ρv and ρe_t).

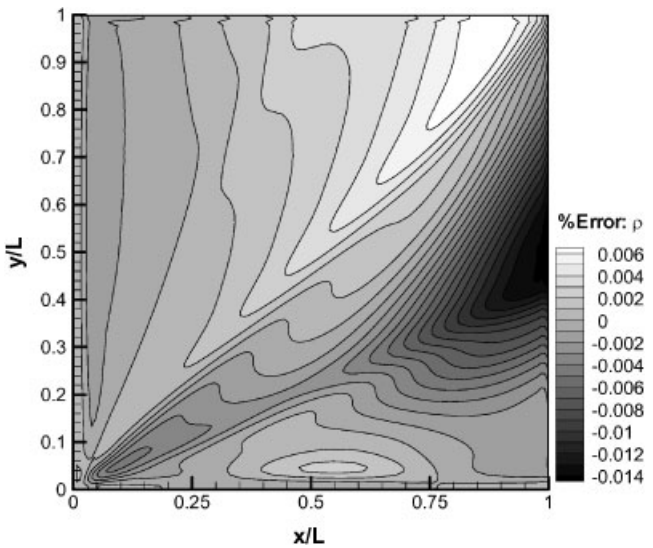

Figure 9. Discretization error in the fine grid Premo solution for density for the Euler case.

The manufactured solutions can also be used to compute local discretization error in the numerical solutions. The discretization error in the finest grid solution using the Premo code is given in Figure 9. The largest errors ($+0.006\%$ and -0.014%) occur at the top and right boundaries, respectively. Since the flow is everywhere supersonic in both the x- and y-directions for this case, the mean flow direction is oriented at a 45° angle to the grid. The error at the bottom left corner both convects downstream and propagates along characteristic Mach lines through the domain.

4.2. Navier–Stokes equations

A manufactured solution was generated for the Navier–Stokes equations (Equations (1), (7) and (8)), along with the auxiliary relationships given in Equations (4)–(6) and (9)–(11). The flow was assumed to be subsonic over the entire domain. The constants used in this manufactured solution are given in Table BII of Appendix B. In order to ensure that the viscous terms were of the same order of magnitude as the convective terms, the absolute viscosity was chosen as $\mu = 10$ N s/m^2. By balancing these two terms, the possibility of a 'false positive' on the order of accuracy test is minimized [8, 20]. If instead, only high Reynolds number solutions were examined (where the diffusive terms were much smaller than the convective terms), then extremely fine meshes would be needed in order to detect discretization mistakes in the viscous terms. These solutions and source terms for mass, momentum and energy are shown graphically in Figures 10 and 11, respectively. Again, both the solutions and the source terms are shown to be smooth, with variations in both the x and y directions.

Iterative convergence histories for both the Wind and Premo codes are given in Figure 12. In order to present results from both codes in the same figure, the number of iterations for

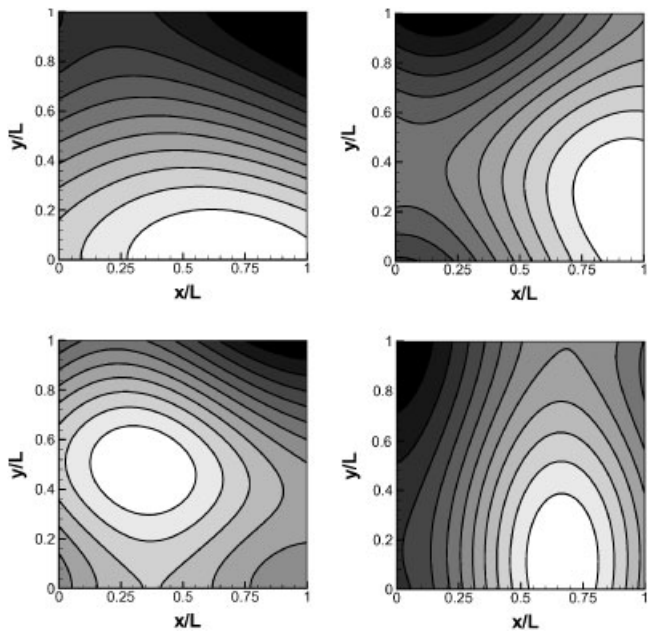

Figure 10. Subsonic Navier–Stokes manufactured solution: ρ (top left), ρe_t (top right), ρu (bottom left) and ρv (bottom right).

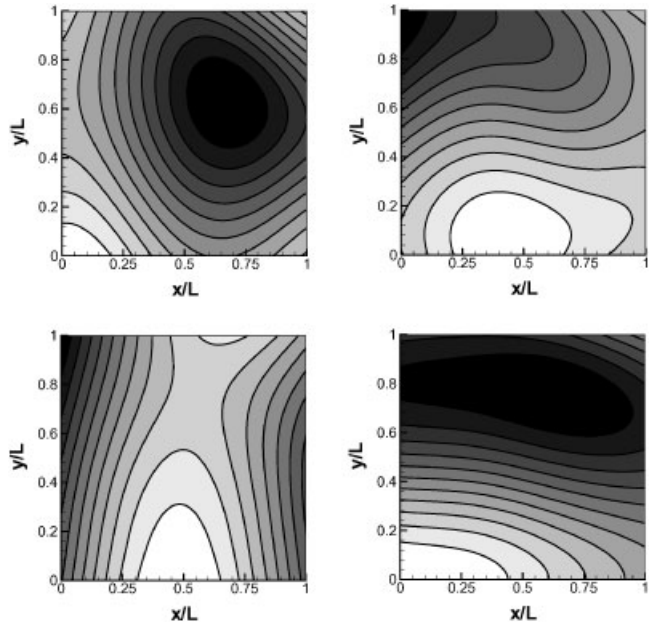

Figure 11. Generated source terms for the subsonic Navier–Stokes case: mass (top left), energy (top right), x-momentum (bottom left), y-momentum (bottom right).

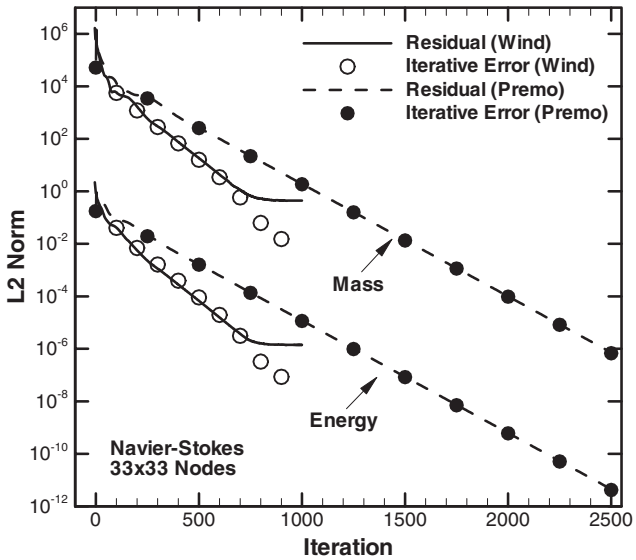

Figure 12. L_2 norms of the steady-state residuals (mass and energy equations) and the iterative error (ρ and ρe_t) for the Navier–Stokes case.

the Premo code, which employs an explicit time integration scheme, were scaled down by a factor of 100. Again, the steady-state residuals for the governing equations (lines) were scaled in order to match the magnitude of the iterative error (symbols). The residuals of the mass and energy equations provide a good indication of the level of iterative error in ρ and ρe_t, respectively. Similar convergence behaviour was observed for momentum (ρu and ρv) and for the other meshes examined. As will be shown, the iterative-error norms are significantly lower than the discretization-error norms.

Discretization-error norms for the conserved variable energy (ρe_t) are given in Figure 13. Owing to the inefficiency of the Runge–Kutta temporal integration scheme, the finest mesh run with the Premo code is the 65×65 node mesh. For both codes, the norms approach the second-order slope as the mesh is refined. These results for spatial convergence are confirmed by examining the observed order of accuracy, shown in Figure 14. Although not shown, a similar behaviour was found for the other conserved variables.

The local discretization error in the energy (ρe_t) is presented in Figure 15 for the fine grid Wind solution. The magnitude of this error is largest at the top boundary ($+0.015\%$) and the right boundary (-0.01%). This error near the boundaries is likely caused by the over-specification of the boundary conditions with the exact Dirichlet values. Recall that, in a one-dimensional inviscid sense, a subsonic inflow requires the specification of two properties and the extrapolation of one property from within the domain, while a subsonic outflow boundary requires the specification of one property and the extrapolation of two properties from within the domain. While the application of a large viscosity value for this manufactured solution makes the use of an inviscid boundary condition questionable, the order of accuracy of the interior points was not affected. Further investigation of appropriate boundary conditions for this case is beyond the scope of this paper.

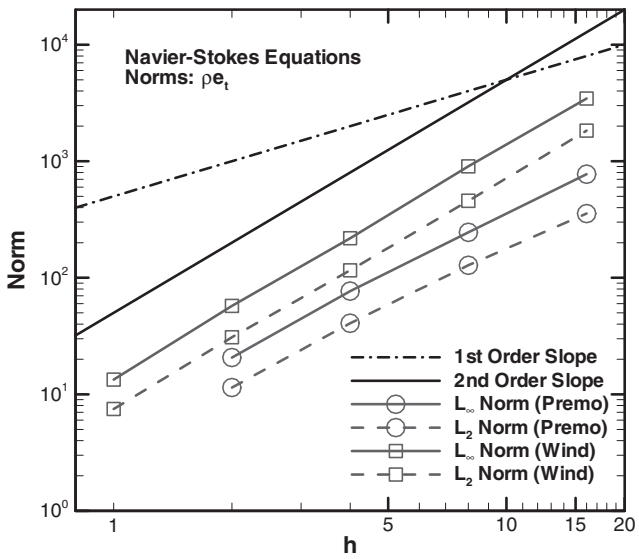

Figure 13. Behaviour of the energy (ρe_t) discretization error norms as the mesh is refined for the Navier–Stokes case.

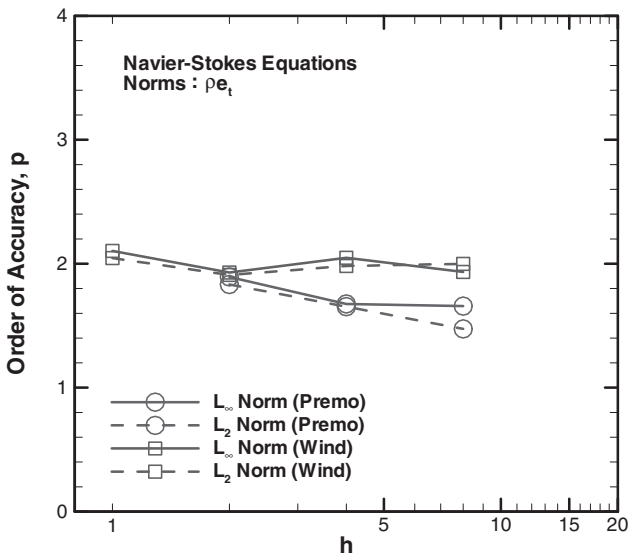

Figure 14. Observed order of accuracy of the energy (ρe_t) discretization error norms as the mesh is refined for the Navier–Stokes case.

5. CONCLUSIONS

The method of manufactured solutions has been applied to the compressible fluid dynamics codes Premo and Wind. Two cases were examined: a supersonic flow governed by the Eu-

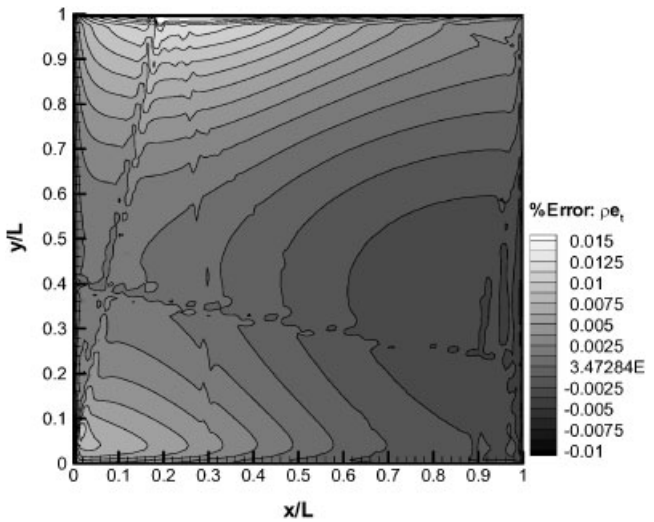

Figure 15. Discretization error in the fine grid Wind solution for energy (ρe_t) for the Navier–Stokes case.

ler equations and a subsonic flow governed by the Navier–Stokes equations. By solving the manufactured solutions on a number of different grid levels, the spatial order of accuracy was ascertained by comparing the numerical solutions to the exact (manufactured) solutions. Both codes demonstrated second-order spatial accuracy as the mesh was refined. By demonstrating that the formal order of accuracy was achieved, the codes were verified, thus providing confidence that there are no mistakes in the spatial discretization for uniform meshes. For the Navier–Stokes case, the specification of inflow and outflow boundary values for each of the primitive variables with exact Dirichlet values resulted in over-specified boundary conditions; however, the order of accuracy was not adversely affected.

The method of manufactured solutions was found to be an invaluable tool for finding coding mistakes. In one case, a coding mistake allowed the gradient term in the MUSCL extrapolation to be zeroed out when the flux limiter (not used in the current work) was set to zero. This error resulted in first-order behaviour of the spatial discretization error which was easily found by the manufactured solutions. As another example, an option for a constant angular rotation was inadvertently activated within the code, which triggered the constant angular velocity terms. This coding mistake was found since non-ordered errors were generated.

A number of coding options were verified by the method of manufactured solutions. The options verified include: inviscid (Euler) and viscous (Navier–Stokes), the Roe upwind scheme (both subsonic and supersonic), the MUSCL extrapolation for second-order convection, the viscous terms and boundary conditions for Dirichlet values and supersonic outflow. Options not verified in the current study include solver efficiency and stability (these are not verifiable with the method), non-uniform or curvilinear meshes, temporal accuracy (the chosen manufactured solutions were not functions of time) and variable transport properties μ and k.

APPENDIX A: ANALYTICAL MANUFACTURED SOLUTION

The form chosen for the manufactured solution for both Euler and Navier–Stokes implementations is given as follows:

$$\rho(x,y) = \rho_0 + \rho_x \sin\left(\frac{a_{\rho x}\pi x}{L}\right) + \rho_y \cos\left(\frac{a_{\rho y}\pi y}{L}\right) + \rho_{xy} \cos\left(\frac{a_{\rho xy}\pi xy}{L^2}\right)$$

$$u(x,y) = u_0 + u_x \sin\left(\frac{a_{ux}\pi x}{L}\right) + u_y \cos\left(\frac{a_{uy}\pi y}{L}\right) + u_{xy} \cos\left(\frac{a_{uxy}\pi xy}{L^2}\right)$$

$$v(x,y) = v_0 + v_x \cos\left(\frac{a_{vx}\pi x}{L}\right) + v_y \sin\left(\frac{a_{vy}\pi y}{L}\right) + v_{xy} \cos\left(\frac{a_{vxy}\pi xy}{L^2}\right)$$

$$p(x,y) = p_0 + p_x \cos\left(\frac{a_{px}\pi x}{L}\right) + p_y \sin\left(\frac{a_{py}\pi y}{L}\right) + p_{xy} \sin\left(\frac{a_{pxy}\pi xy}{L^2}\right) \tag{A1}$$

APPENDIX B: MANUFACTURED SOLUTION CONSTANTS

Constants employed for the manufactured solutions include $L = 1\text{m}$, $\gamma = 1.4$ and $R = 287.0\text{Nm/(kg K)}$. For the Navier–Stokes calculations, additional constants include the absolute viscosity $\mu = 10\,\text{N s/m}^2$ and the Prandtl number $Pr = 1.0$. The constants for the supersonic Euler manufactured solution are given in Table BI, and the constants for the subsonic Navier–Stokes manufactured solution are given in Table BII. Note that the ϕ constants all have the same dimensions as the primitive variable (listed in the first column), and the a constants are dimensionless.

Table BI. Constants for supersonic Euler manufactured solution.

Equation, ϕ	ϕ_0	ϕ_x	ϕ_y	ϕ_{xy}	$a_{\phi x}$	$a_{\phi y}$	$a_{\phi xy}$
$\rho(\text{kg/m}^3)$	1	0.15	−0.1	0	1	0.5	0
$u(\text{m/s})$	800	50	−30	0	1.5	0.6	0
$v(\text{m/s})$	800	−75	40	0	0.5	2./3	0
$p(\text{N/m}^2)$	1×10^5	0.2×10^5	0.5×10^5	0	2	1	0

Table BII. Constants for subsonic Navier–Stokes manufactured solution.

Equation, ϕ	ϕ_0	ϕ_x	ϕ_y	ϕ_{xy}	$a_{\phi x}$	$a_{\phi y}$	$a_{\phi xy}$
$\rho(\text{kg/m}^3)$	1	0.1	0.15	0.08	0.75	1.0	1.25
$u(\text{m/s})$	70	4	−12	7	5./3	1.5	0.6
$v(\text{m/s})$	90	−20	4	−11	1.5	1.0	0.9
$p(\text{N/m}^2)$	1×10^5	-0.3×10^5	0.2×10^5	-0.25×10^5	1.0	1.25	0.75

APPENDIX C: EXAMPLE SOURCE TERM

A sample source term is given in Equation (1) for the mass conservation equation. This source terms is the result of analytically differentiating the general manufactured solution given in Appendix A for ρ, u and v according to the mass conservation equation. These manufactured solutions can be obtained from the first author:

$$\begin{aligned}
\frac{\partial(\rho)}{\partial t} + \frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} &= f_m \\
&= \left[v_0 + v_x \cos\left(\frac{a_{vx}\pi x}{L}\right) + v_y \sin\left(\frac{a_{vy}\pi y}{L}\right) + v_{xy} \cos\left(\frac{a_{vxy}\pi xy}{L^2}\right)\right] \\
&\quad \times \left[-\frac{a_{\rho y}\pi\rho_y}{L} \sin\left(\frac{a_{\rho y}\pi y}{L}\right) - \frac{a_{\rho xy}\pi\rho_{xy}x}{L^2} \sin\left(\frac{a_{\rho xy}\pi xy}{L^2}\right)\right] \\
&\quad + \left[u_0 + u_x \sin\left(\frac{a_{ux}\pi x}{L}\right) + u_y \cos\left(\frac{a_{uy}\pi y}{L}\right) + u_{xy} \cos\left(\frac{a_{uxy}\pi xy}{L^2}\right)\right] \\
&\quad \times \left[\frac{a_{\rho x}\pi\rho_x}{L} \cos\left(\frac{a_{\rho x}\pi x}{L}\right) - \frac{a_{\rho xy}\pi\rho_{xy}y}{L^2} \sin\left(\frac{a_{\rho xy}\pi xy}{L^2}\right)\right] \\
&\quad + \left[\rho_0 + \rho_x \sin\left(\frac{a_{\rho x}\pi x}{L}\right) + \rho_y \cos\left(\frac{a_{\rho y}\pi y}{L}\right) + \rho_{xy} \cos\left(\frac{a_{\rho xy}\pi xy}{L^2}\right)\right] \\
&\quad \times \left[\frac{a_{ux}\pi u_x}{L} \cos\left(\frac{a_{ux}\pi x}{L}\right) - \frac{a_{uxy}\pi u_{xy}y}{L^2} \sin\left(\frac{a_{uxy}\pi xy}{L^2}\right)\right] \\
&\quad + \left[\rho_0 + \rho_x \sin\left(\frac{a_{\rho x}\pi x}{L}\right) + \rho_y \cos\left(\frac{a_{\rho y}\pi y}{L}\right) + \rho_{xy} \cos\left(\frac{a_{\rho xy}\pi xy}{L^2}\right)\right] \\
&\quad \times \left[\frac{a_{vy}\pi v_y}{L} \cos\left(\frac{a_{vy}\pi y}{L}\right) - \frac{a_{vxy}\pi v_{xy}x}{L^2} \sin\left(\frac{a_{vxy}\pi xy}{L^2}\right)\right] \qquad \text{(C1)}
\end{aligned}$$

ACKNOWLEDGEMENTS

The authors would like to thank Patrick Knupp and Stefan Domino of Sandia National Laboratories for their helpful reviews of this manuscript.

REFERENCES

1. Roache PJ. *Verification and Validation in Computational Science and Engineering*. Hermosa Publishers: New Mexico, 1998.
2. *Guide for the Verification and Validation of Computational Fluid Dynamics Simulations*. AIAA G-077-1998; 3.
3. Knupp P, Salari K. In *Verification of Computer Codes in Computational Science and Engineering*, Rosen KH (ed.). Chapman and Hall/CRC: Boca Raton, FL, 2003.
4. Roache PJ, Steinberg S. Symbolic manipulation and computational fluid dynamics. *AIAA Journal* 1984; **22**(10):1390–1394.
5. Roache PJ, Knupp PM, Steinberg S, Blaine RL. Experience with benchmark test cases for groundwater flow. In *Benchmark Test Cases for Computational Fluid Dynamics*, Celik I, Freitas CJ (eds). ASME FED 93, Book No. H00598. 1990; 49–56.

6. Oberkampf WL, Blottner FG. Issues in computational fluid dynamics code verification and validation. *AIAA Journal* 1998; **36**(5):687–695 (see also Oberkampf WL, Blottner FG, Aeschliman DP. Methodology for computational fluid dynamics code verification/validation. *AIAA Paper* 95-2226, 1995).
7. Salari K, Knupp P. Code verification by the method of manufactured solutions. *SAND* 2000-1444, Sandia National Laboratories, Albuquerque, NM, June 2000.
8. Roache PJ. Code verification by the method of manufactured solutions. *Journal of Fluids Engineering* 2002; **124**(1):4–10.
9. Edwards HC, Stewart JR. SIERRA: a software environment for developing complex multiphysics applications. In *Proceedings of the First MIT Conference on Computational Fluid and Solid Mechanics*, Bathe KJ (ed.). Elsevier Scientific: Cambridge, MA, June 2001.
10. Nelson CC, Power GD. CHSSI project CFD-7: the NPARC alliance flow simulation system. *AIAA Paper* 2001-0594, 2001.
11. Smith TM, Ober CC, Lorber AA. SIERRA/Premo—a new general purpose compressible flow simulation code. *AIAA Paper* 2002-3292, 2002.
12. Roe PL. Approximate Riemann solvers, parameter vectors and difference schemes. *Journal of Computational Physics* 1981; **43**(2):357–372.
13. van Leer B. Towards the ultimate conservative difference scheme III. Upstream-centered finite-difference schemes for ideal compressible flow. *Journal of Computational Physics* 1977; **23**(3):263–275.
14. Haselbacher A, Blazek J. Accurate and efficient discretization of Navier–Stokes equations on mixed grids. *AIAA Journal* 2000; **38**(11):2094–2102.
15. Fromm JE. A method for reducing dispersion in convective difference schemes. *Journal of Computational Physics* 1968; **3**:176–189.
16. van der Houwen PJ. Explicit Runge–Kutta formulas with increased stability boundaries. *Numerische Mathematik* 1972; **20**:149–164.
17. Bush RH. A three dimensional zonal Navier–Stokes code for subsonic through hypersonic propulsion flowfields. *AIAA Paper* 88-2830, 1988.
18. Bush RH, Power GD, Towne CE. WIND: the production flow solver of the NPARC alliance. *AIAA Paper* 98-0935, 1998.
19. Power GD, Underwood ML. Wind 2.0—Progress on an applications-oriented CFD code. *AIAA Paper* 99-3212, 1999.
20. Roy CJ, Smith TM, Ober CC. Verification of a compressible CFD code using the method of manufactured solutions. *AIAA Paper* 2002-3110, 2002.
21. Oberkampf WL, Trucano TG. Verification and validation in computational fluid dynamics. *Progress in Aerospace Sciences* 2002; **38**:209–272.

