6

Exact solutions

The primary focus of this chapter is on the use of exact solutions to mathematical models for code verification. Recall that, in some cases, software testing can be performed by simply running the code and comparing the results to the correct code output. However, in scientific computing, "correct" code output depends on the chosen spatial mesh, time step, iterative convergence tolerance, machine precision, etc. We are thus forced to rely on other less definitive methods for assessing code correctness. In Chapter 5, the order of accuracy test was argued to be the most rigorous approach for code verification. When the order of accuracy test fails, or when the formal order of accuracy has not been determined, then the less rigorous convergence test may be used. In either case, an exact solution to the underlying mathematical model is required. When used for code verification, the ability of this exact solution to exercise all terms in the mathematical model is more important than any physical realism of the solution. In fact, realistic exact solutions are often avoided for code verification due to the presence of singularities and/or discontinuities. Numerous examples will be given in this chapter of exact solutions and their use with the order verification test. The final example given in this chapter employs the less rigorous convergence test with benchmark numerical solutions. In addition to code verification applications, exact solutions to mathematical models are extremely valuable for evaluating the accuracy of numerical schemes, determining solution sensitivity to mesh quality and topology, evaluating the reliability of discretization error estimators, and evaluating solution adaptation schemes. For these secondary applications, physically realistic exact solutions are preferred (see Section 6.4).

This chapter discusses methods for obtaining exact solutions to mathematical models used in scientific computing. These mathematical models typically take the form of either integral or differential equations. Because we will have to compute actual numerical values for these exact solutions, we will use a different definition for an exact solution than found in standard mathematics textbooks. The definition used here for an exact solution is a solution to the mathematical model that is in closed form, i.e., in terms of elementary functions (trigonometric functions, exponentials, logarithms, powers, etc.) or readily-computed special functions (e.g., gamma, beta, error, Bessel functions) of the independent variables. Solutions involving infinite series or a reduction of a partial differential equation to a system of ordinary differential equations which do not have exact solutions will be considered as approximate solutions and are addressed at the end of the chapter.

In scientific computing, the mathematical models can take different forms. When a set of physical laws such as conservation of mass, momentum, or energy (i.e., the conceptual model) are formulated for an infinitesimally small region of space, then the resulting mathematical model generally takes the form of differential equations. When applied to a region of space with finite size, the mathematical model takes the form of integral (or integro-differential) equations. The differential form is called the *strong form* of the equations, whereas the integral form, after application of the divergence theorem (which converts the volume integral of the gradient of a quantity to fluxes through the boundary), is called the *weak form*. The finite difference method employs the strong form of the equations, while the finite element and finite volume methods employ the weak form.

The strong form explicitly requires solutions that are differentiable, whereas the weak form admits solutions that can contain discontinuities while still satisfying the underlying physical laws, and these discontinuous solutions are called *weak solutions*. Weak solutions satisfy the differential equation only in a restricted sense. While exact solutions to the weak form of the equations exist that do contain discontinuities (e.g., the Riemann or shock tube problem in gas dynamics), the more general approaches for generating exact solutions such as the method of manufactured solutions discussed in Section 6.3 have not to our knowledge encompassed nondifferentiable weak solutions (although this extension is needed). This chapter will assume the strong (i.e., differential) form of the mathematical models unless otherwise noted. Since strong solutions also satisfy the weak form of the equations, the finite element and finite volume methods will not be excluded by this assumption, and we are simply restricting ourselves to smooth solutions.

Because scientific computing often involves complex systems of coupled partial differential equations (PDEs) which have relatively few exact solutions, the organization of this chapter is very different from that of standard mathematics texts. After a short introduction to differential equations in Section 6.1, a discussion of "traditional" exact solutions and solution methods is presented in Section 6.2. The method of manufactured solutions (MMS) is a more general approach for obtaining exact solutions to complicated mathematical models and is discussed in detail in Section 6.3. When physically realistic manufactured solutions are desired, the approaches discussed in Section 6.4 can be used. As discussed above, solutions involving infinite series, reduction of PDEs to ordinary differential equations, or numerical solutions of the underlying mathematical model with established numerical accuracy are relegated to Section 6.5.

6.1 Introduction to differential equations

Differential equations are ubiquitous in the study of physical processes in science and engineering (O'Neil, 2003). A differential equation is a relation between a variable and its derivatives. When only one independent variable is present, the equation is called an *ordinary differential equation*. A differential equation involving derivatives with respect to two or more independent variables (e.g., x and t, or x, y, and z) is called a *partial differential equation* (PDE). A differential equation can be a single equation with a single

dependent variable or a system of equations with multiple dependent variables. The *order* of a differential equation is the largest number of derivatives applied to any dependent variable. The *degree* of a differential equation is the highest power of the highest derivative found in the equation.

A differential equation is considered to be *linear* when the dependent variables and all of their derivatives occur with powers less than or equal to one and there are no products involving derivatives and/or functions of the dependent variables. Solutions to linear differential equations can be combined to form new solutions using the linear superposition principle. A *quasi-linear* differential equation is one that is linear in the highest derivative, i.e., the highest derivative appears to the power one.

General solutions to PDEs are solutions that that satisfy the PDE but involve arbitrary constants and/or functions and thus are not unique. To find *particular solutions* to PDEs, additional conditions must be supplied on the boundary of the domain of interest, i.e., *boundary conditions*, at an initial data location i.e., *initial conditions*, or some combination of the two. Boundary conditions generally come in the form of *Dirichlet* boundary conditions which specify values of the dependent variables or *Neumann* boundary conditions which specify the values of derivatives of the dependent variables normal to the boundary. When both Dirichlet and Neumann conditions are applied at a boundary it is called a *Cauchy* boundary condition, whereas a linear combination of a dependent variable and its normal derivative is called a *Robin* boundary condition. The latter is often confused with *mixed* boundary conditions, which occur when different boundary condition types (Dirichlet, Neumann, or Robin) are applied at different boundaries in a given problem.

Another source of confusion is related to the order of the boundary conditions. The maximum order of the boundary conditions is at least one less than the order of the differential equation. Here *order* refers to the highest number of derivatives applied to any dependent variable as discussed above. This requirement on the order of the boundary condition is sometimes erroneously stated as a requirement on the order of accuracy of the discretization of a derivative boundary condition when the PDE is solved numerically. On the contrary, a reduction in the formal order of accuracy of a discretized boundary condition often leads to a reduction in the observed order of accuracy of the entire solution.

Partial differential equations can be classified as *elliptic*, *parabolic*, *hyperbolic*, or a combination of these types. For scalar equations, this classification is fairly straightforward in a manner analogous to determining the character of algebraic equations. For systems of PDEs written in quasi-linear form, the eigenvalues of the coefficient matrices can be used to determine the mathematical character (Hirsch, 2007).

6.2 Traditional exact solutions

The standard approach to obtaining an exact solution to a mathematical model can be summarized as follows. Given the governing partial differential (or integral) equations on some domain with appropriately specified initial and/or boundary conditions, find the exact solution. The main disadvantage of this approach is that there are only a limited number

of exact solutions known for complex equations. Here the complexity of the equations could be due to geometry, nonlinearity, physical models, and/or coupling between multiple physical phenomena such as fluid–structure interaction.

When exact solutions are found for complex equations, they often depend on significant simplifications in dimensionality, geometry, physics, etc. For example, the flow between infinite parallel plates separated by a small gap with one plate moving at a constant speed is called Couette flow and is described by the Navier–Stokes equations, a nonlinear, second-order system of PDEs. In Couette flow, the velocity profile is linear across the gap, and this linearity causes the diffusion term, a second derivative of velocity, to be identically zero. In addition, there are no solution variations in the direction that the plate is moving. Thus the exact solution to Couette flow does not exercise many of the terms in the Navier–Stokes equations.

There are many books available that catalogue a vast number of exact solutions for differential equations found in science and engineering. These texts address ordinary differential equations (e.g., Polyanin and Zaitsev, 2003), linear PDEs (Kevorkian, 2000; Polyanin, 2002; Meleshko, 2005), and nonlinear PDEs (Kevorkian, 2000; Polyanin and Zaitsev, 2004; Meleshko, 2005). In addition, many exact solutions can be found in discipline-specific references such as those for heat conduction (Carslaw and Jaeger, 1959), fluid dynamics (Panton, 2005; White, 2006), linear elasticity (Timoshenko and Goodier, 1969; Slaughter, 2002), elastodynamics (Kausel, 2006), and vibration and buckling (Elishakoff, 2004). The general Riemann problem involves an exact weak (i.e., discontinuous) solution to the 1-D unsteady inviscid equations for gas dynamics (Gottlieb and Groth, 1988).

6.2.1 Procedures

In contrast to the method of manufactured solutions discussed in Section 6.3, the traditional method for finding exact solutions solves the forward problem: given a partial differential equation, a domain, and boundary and/or initial conditions, find the exact solution. In this section, we present some of the simpler classical methods for obtaining exact solutions and make a brief mention of more advanced (nonclassical) techniques. Further details on the classical techniques for PDEs can be found in Ames (1965) and Kevorkian (2000).

6.2.1.1 Separation of variables

Separation of variables is the most common approach for solving linear PDEs, although it can also be used to solve certain nonlinear PDEs. Consider a scalar PDE with dependent variable u and independent variables t and x. There are two forms for separation of variables, multiplicative and additive, and these approaches can be summarized as:

$$\text{multiplicative: } u(t, x) = \phi(t)\psi(x),$$

$$\text{additive: } u(t, x) = \phi(t) + \psi(x).$$

The multiplicative form of separation of variables is the most common.

For an example of separation of variables, consider the 1-D unsteady heat equation with constant thermal diffusivity α,

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}, \tag{6.1}$$

where $T(t, x)$ is the temperature. Let us first simplify this equation by employing the simple transformations $t = \alpha \bar{t}$ and $x = \alpha \bar{x}$. With these transformations, the heat equation can be rewritten in simpler form as:

$$\frac{\partial T}{\partial \bar{t}} = \frac{\partial^2 T}{\partial \bar{x}^2}. \tag{6.2}$$

Using the multiplicative form of separation of variables $T(\bar{x}, \bar{t}) = \phi(\bar{t})\psi(\bar{x})$, the differential equation can be rewritten as

$$\frac{\phi_t(\bar{t})}{\phi(\bar{t})} = \frac{\psi_{xx}(\bar{x})}{\psi(\bar{x})}, \tag{6.3}$$

where the subscript denotes differentiation with respect to the subscripted variable. Since the left hand side of Eq. (6.3) is independent of $\bar{x}$ and the right hand side is independent of $\bar{t}$, both sides must be equal to a constant a, i.e.,

$$\frac{\phi_t(\bar{t})}{\phi(\bar{t})} = \frac{\psi_{xx}(\bar{x})}{\psi(\bar{x})} = a. \tag{6.4}$$

Each side of Eq. (6.3) can thus be written as

$$\begin{aligned} \frac{\mathrm{d}\phi}{\mathrm{d}\bar{t}} - a\phi &= 0, \\ \frac{\mathrm{d}^2\psi}{d\bar{x}^2} - a\psi &= 0. \end{aligned} \tag{6.5}$$

Equations (6.5) can be integrated using standard methods for ordinary differential equations. After substituting back in for x and t, we finally arrive at two general solutions (Meleshko, 2005) depending on the sign of a:

$$\begin{aligned} a = \lambda^2 &: u(t, x) = \exp\left(\frac{\lambda^2 t}{\alpha}\right)\left[c_1 \exp\left(\frac{-\lambda x}{\alpha}\right) + c_2 \exp\left(\frac{\lambda x}{\alpha}\right)\right], \\ a = -\lambda^2 &: u(t, x) = \exp\left(\frac{-\lambda^2 t}{\alpha}\right)\left[c_1 \sin\left(\frac{\lambda x}{\alpha}\right) + c_2 \cos\left(\frac{\lambda x}{\alpha}\right)\right], \end{aligned} \tag{6.6}$$

where c_1, c_2, and λ are constants that can be determined from the initial and boundary conditions.

6.2.1.2 Transformations

Transformations can sometimes be used to convert a differential equation into a simpler form that has a known solution. Transformations that do not involve derivatives are called point transformations (Polyanin and Zaitsev, 2004), while transformations that involve

derivatives are called tangent transformations (Meleshko, 2005). An example of a point transformation is the hodograph transformation, which exchanges the roles between the independent and the dependent variables. Examples of tangent transformations include Legendre, Hopf–Cole, and Laplace transformations.

A well-known example of a tangent transformation is the Hopf–Cole transformation (Polyanin and Zaitsev, 2004). Consider the nonlinear Burgers' equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}, \tag{6.7}$$

where the viscosity ν is assumed to be constant. Burgers' equation serves as a scalar model equation for the Navier–Stokes equations since it includes an unsteady term $\left(\frac{\partial u}{\partial t}\right)$, a nonlinear convection term $\left(u\frac{\partial u}{\partial x}\right)$, and a diffusion term $\left(\nu\frac{\partial^2 u}{\partial x^2}\right)$. The Hopf–Cole transformation is given by

$$u = \frac{-2\nu}{\phi}\phi_x, \tag{6.8}$$

where again ϕ_x denotes partial differentiation of ϕ with respect to x. Substituting the Hopf–Cole transformation into Burgers' equation, applying the product rule, and simplifying results in

$$\frac{\phi_{tx}}{\phi} - \frac{\phi_t\phi_x}{\phi^2} - \nu\left(\frac{\phi_{xxx}}{\phi} - \frac{\phi_{xx}\phi_x}{\phi^2}\right) = 0, \tag{6.9}$$

which can be rewritten as

$$\frac{\partial}{\partial x}\left[\frac{1}{\phi}\left(\frac{\partial\phi}{\partial t} - \nu\frac{\partial^2\phi}{\partial x^2}\right)\right] = 0. \tag{6.10}$$

The terms in parenthesis in Eq. (6.10) are simply the 1-D unsteady heat equation (6.1) written with $\phi(t, x)$ as the dependent variable. Thus any nonzero solution to the heat equation $\phi(t, x)$ can be transformed into a solution to Burgers' equation (6.7) using the Hopf–Cole transformation given by Eq. (6.8).

6.2.1.3 Method of characteristics

An approach for finding exact solutions to hyperbolic PDEs is the *method of characteristics*. The goal is to identify characteristic curves/surfaces along which certain solution properties will be constant. Along these characteristics, the PDE can be converted into a system of ordinary differential equations which can be integrated by starting at an initial data location. When the resulting ordinary differential equations can be solved analytically in closed form, then we will consider the solution to be an exact solution, whereas solutions requiring numerical integration or series solutions will be considered approximate (see Section 6.5).

Table 6.1 *Exact solutions to the 1-D unsteady heat conduction equation.*

Solutions to $\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2}$
$T(t,x) = A\left(x^3 + 6\alpha t x\right) + B$
$T(t,x) = A\left(x^4 + 12\alpha t x^2 + 12\alpha^2 t^2\right) + B$
$T(t,x) = x^{2n} + \sum_{k=1}^{n} \frac{(2n)(2n-1)\cdots(2n-2k+1)}{k!}(\alpha t)^k x^{2n-2k}$
$T(t,x) = A\exp\left(\alpha\mu^2 t \pm \mu x\right) + B$
$T(t,x) = A\frac{1}{\sqrt{t}}\exp\left(\frac{-x^2}{4\alpha t}\right) + B$
$T(t,x) = A\exp\left(-\mu x\right)\cos\left(\mu x - 2\alpha\mu^2 t + B\right) + C$
$T(t,x) = A\,\mathrm{erf}\left(\frac{x}{2\sqrt{\alpha t}}\right) + B$

6.2.1.4 *Advanced approaches*

Additional approaches for obtaining exact solutions to PDEs were developed in the latter half of the twentieth century. One example is the method of differential constraints developed by Yanenko (1964). Another example is the application of group theory, which has found extensive applications in algebra and geometry, for finding solutions to differential equations. While a discussion of these nonclassical analytic solutions techniques is beyond the scope of this book, additional information on the application of these methods for PDEs can be found in Polyanin (2002), Polyanin and Zaitsev (2004), and Meleshko (2005).

6.2.2 *Example exact solution: 1-D unsteady heat conduction*

Some general solutions to the 1-D unsteady heat conduction equation given by Eq. (6.1) are presented in Table 6.1 above, where A, B, C, and μ are arbitrary constants and n is a positive integer. These solutions, as well as many others, can be found in Polyanin (2002). Employing Eq. (6.8), these solutions can also be transformed into solutions to Burgers' equation.

6.2.3 *Example with order verification: steady Burgers' equation*

This section describes an exact solution for the steady Burgers' equation, which is then employed in an order verification test for a finite difference discretization. Benton and Platzmann (1972) describe 35 exact solutions to Burgers' equation, which is given above in Eq. (6.7). The steady-state form of Burgers' equation is found by restricting the solution u to be a function of x only, thus reducing Eq. (6.7) to the following ordinary differential

equation

$$u\frac{\mathrm{d}u}{\mathrm{d}x} = \nu\frac{\mathrm{d}^2u}{\mathrm{d}x^2}, \tag{6.11}$$

where u is a velocity and ν is the viscosity. The exact solution to Burgers' equation for a steady, viscous shock (Benton and Platzmann, 1972) is given in dimensionless form, denoted by primes, as

$$u'(x') = -2\tanh(x'). \tag{6.12}$$

This dimensionless solution for Burgers' equation can be converted to dimensional quantities via transformations given by $x' = x/L$ and $u' = uL/\nu$ where L is a reference length scale. This solution for Burgers' equation is also invariant to scaling by a factor α as follows: $\bar{x} = x/\alpha$ and $\bar{u} = \alpha u$. Finally, one can define a Reynolds number in terms of L, a reference velocity u_{ref}, and the viscosity ν as

$$\mathrm{Re} = \frac{u_{\mathrm{ref}}L}{\nu}, \tag{6.13}$$

where the domain is generally chosen as $-L \le x \le L$ and u_{ref} the maximum value of u on the domain.

For this example, the steady Burgers' equation is discretized using the simple implicit finite difference scheme given by

$$\bar{u}_i\left(\frac{u_{i+1}^{k+1} - u_{i-1}^{k+1}}{2\Delta x}\right) - \nu\left(\frac{u_{i+1}^{k+1} - 2u_i^{k+1} + u_{i-1}^{k+1}}{\Delta x^2}\right) = 0, \tag{6.14}$$

where a uniform mesh with spacing Δx is used between the spatial nodes which are indexed by i. The formal order of accuracy of this discretization scheme is two and can be found from a truncation error analysis. The above discretization scheme is linearized by setting $\bar{u}_i = u_i^k$ and then iterated until the solution at iteration k satisfies Eq. (6.14) within round-off error using double precision computations. This results in the iterative residuals, found by substituting $\bar{u}_i = u_i^{k+1}$ into Eq. (6.14), being reduced by approximately fourteen orders of magnitude. Thus, iterative convergence and round-off errors can be neglected (see Chapter 7), and the numerical solutions effectively contain only discretization error.

The solution to Burgers' equation for a Reynolds number of eight is given in Figure 6.1, which includes both the exact solution (in scaled dimensional variables) and a numerical solution obtained using 17 evenly-spaced points (i.e., nodes). The low value for the Reynolds number was chosen for this code verification exercise to ensure that both the convection and the diffusion terms are exercised. Note that choosing a large Reynolds number effectively scales down the diffusion term since it is multiplied by 1/Re when written in dimensionless form. For higher Reynolds numbers, extremely fine meshes would be needed to detect coding mistakes in the diffusion term.

Numerical solutions for the steady Burgers' equation were obtained on seven uniform meshes from the finest mesh of 513 nodes ($h = 1$) to the coarsest mesh of nine nodes

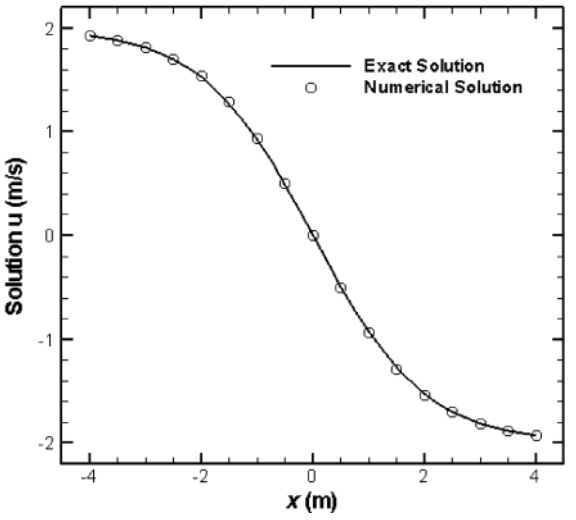

Figure 6.1 Comparison of the numerical solution using 17 nodes with exact solution for the steady Burgers equation with Reynolds number Re = 8.

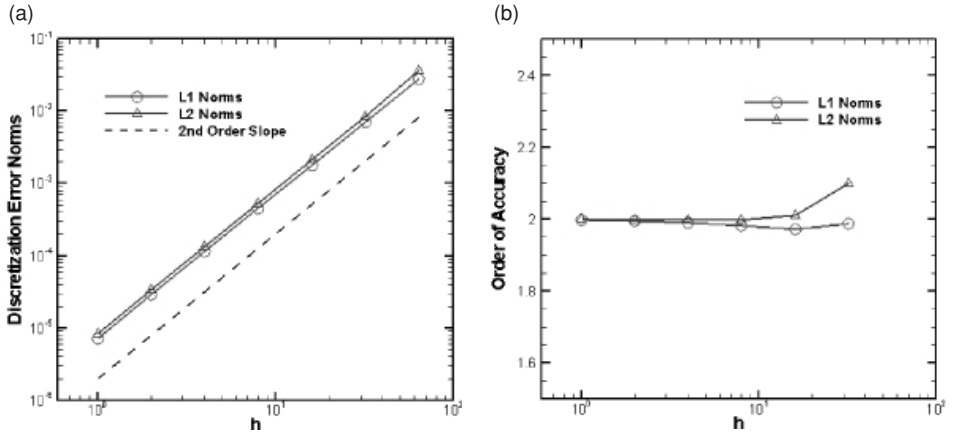

Figure 6.2 Discretization error (a) and observed orders of accuracy (b) for steady Burger's equation with Reynolds number Re = 8.

($h = 64$). Discrete L_1 and L_2 norms of the discretization error are given in Figure 6.2a and both norms appear to reduce with mesh refinement at a second-order rate. The order of accuracy, as computed from Eq. (5.23), is given in Figure 6.2b. Both norms rapidly approach the scheme's formal order of accuracy of two with mesh refinement for this simple problem. The code used to compute the numerical solutions to Burgers' equation is thus considered to be verified for the options exercised, namely steady-state solutions on a uniform mesh.

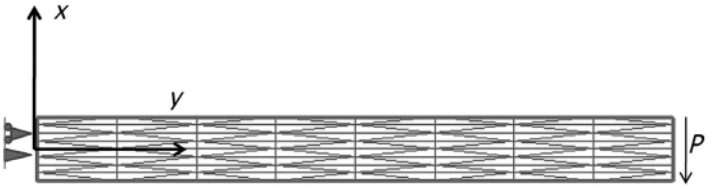

Figure 6.3 2-D linear elasticity problem for plane stress in a cantilevered beam loaded at the tip; an unstructured mesh with 64 triangular elements is also shown.

6.2.4 Example with order verification: linear elasticity

This section describes an exact solution for linear elasticity and includes an example of order verification for a finite element code. The problem of interest is a cantilevered beam that is loaded at the tip, as shown in Figure 6.3. The equations governing the displacements u and v in the x and y directions, respectively, arise from the static equilibrium linear momentum equations for plane stress. An isotropic linear elastic material is assumed along with small strain (i.e., small deformation gradient) assumptions. The equations governing the displacements can be written as

$$\begin{aligned}\left(1+\frac{1-\alpha}{2\alpha-1}\right)\frac{\partial^2 u}{\partial x^2}+\frac{1}{2}\frac{\partial^2 u}{\partial y^2}+\left(\frac{1}{2}+\frac{1-\alpha}{2\alpha-1}\right)\frac{\partial^2 v}{\partial x\partial y}=0,\\ \left(1+\frac{1-\alpha}{2\alpha-1}\right)\frac{\partial^2 v}{\partial y^2}+\frac{1}{2}\frac{\partial^2 v}{\partial x^2}+\left(\frac{1}{2}+\frac{1-\alpha}{2\alpha-1}\right)\frac{\partial^2 u}{\partial x\partial y}=0,\end{aligned} \tag{6.15}$$

where for plane stress

$$\alpha=\frac{1}{1+\nu}$$

and ν is Poisson's ratio.

For a beam of length L, height h, and width (in the z direction) of w, an exact solution can be found for the displacements (Slaughter, 2002). This solution has been modified (Seidel, 2009) using the above coordinate system resulting in an Airy stress function of

$$\Phi=-2\frac{xPy^3}{h^3w}+2\frac{LPy^3}{h^3w}+3/2\frac{xyP}{hw}, \tag{6.16}$$

which exactly satisfies the equilibrium and compatibility conditions. The stresses can then be easily obtained by

$$\begin{aligned}\sigma_{xx}&=\frac{\partial^2\Phi}{\partial y^2}=-12\frac{xyP}{h^3w}+12\frac{LPy}{h^3w},\\ \sigma_{yy}&=\frac{\partial^2\Phi}{\partial x^2}=0,\\ \sigma_{xy}&=\frac{\partial^2\Phi}{\partial x\partial y}=6\frac{Py^2}{h^3w}-3/2\frac{P}{hw}.\end{aligned} \tag{6.17}$$

The stress–strain relationship is given by

$$\sigma_{xx} = \frac{E}{1-\nu^2}\left(\varepsilon_{xx} + \nu\varepsilon_{yy}\right),$$
$$\sigma_{yy} = \frac{E}{1-\nu^2}\left(\nu\varepsilon_{xx} + \varepsilon_{yy}\right),$$
$$\sigma_{xy} = \frac{E}{1+\nu}\varepsilon_{xy},$$

and the strain is related to the displacements by

$$\varepsilon_{xx} = \frac{\partial u}{\partial x},$$
$$\varepsilon_{yy} = \frac{\partial v}{\partial y},$$
$$\varepsilon_{xy} = \frac{1}{2}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right).$$

This solution results in traction-free conditions on the upper and lower surfaces, i.e.,

$$\sigma_{yy}(x, h/2) = \sigma_{xy}(x, h/2) = \sigma_{yy}(x, -h/2) = \sigma_{xy}(x, -h/2) = 0,$$

and static equivalent tip loads of zero net axial force, zero bending moment, and the applied shear force at the tip of $-P$:

$$\int_{-h/2}^{h/2} \sigma_{xx}(L, y)\mathrm{d}y = 0,$$
$$\int_{-h/2}^{h/2} y\sigma_{xx}(L, y)\mathrm{d}y = 0,$$
$$\int_{-h/2}^{h/2} \sigma_{xy}(L, y)\mathrm{d}y = -P/w.$$

The conditions at the wall are fully constrained at the neutral axis ($y = 0$) and no rotation at the top corner ($y = h/2$):

$$u(0, 0) = 0,$$
$$v(0, 0) = 0,$$
$$u(0, h/2) = 0.$$

The displacement in the x and y directions thus become

$$u = 1/2\left(2\frac{Py^3}{h^3 w} - 3/2\frac{yP}{hw} + \alpha\left(-6\frac{x^2 Py}{h^3 w} + 12\frac{LPyx}{h^3 w} + 2\frac{Py^3}{h^3 w} - 1/2\frac{(-2P + \alpha P)y}{w\alpha h}\right)\right)\mu^{-1},$$

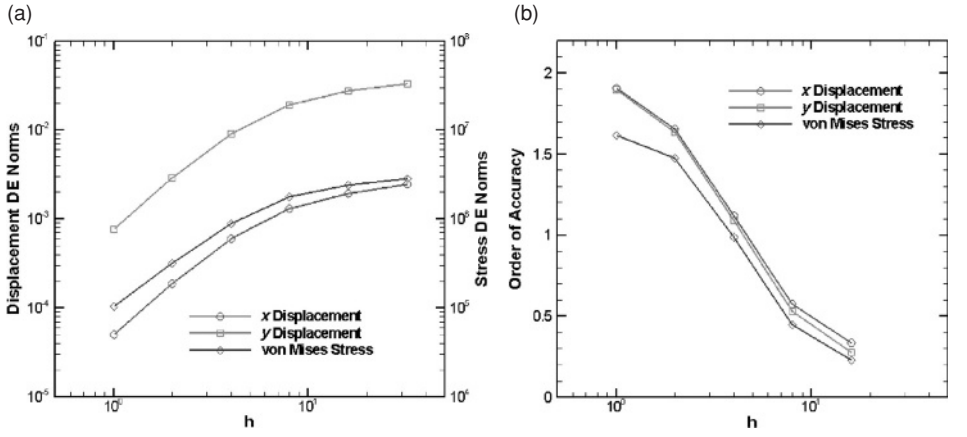

Figure 6.4 Discretization error (a) and observed orders of accuracy (b) for the steady Burger's equation with Reynolds number Re = 8.

$$v = 1/2\left(6\frac{xPy^2}{h^3w} - 6\frac{xPLy^2}{h^3w} - 3/2\frac{xP}{hw} + \alpha\left(-6\frac{xPy^2}{h^3w} + 6\frac{PLy^2}{h^3w} + 2\frac{x^3P}{h^3w} - 6\frac{PLx^2}{h^3w} + 1/2\frac{(-2P+\alpha P)x}{w\alpha h}\right)\right)\mu^{-1}, \tag{6.18}$$

where μ is the shear modulus.

Finite element solutions were obtained for the weak form of Eq. (6.15) using linear basis functions, which gives a formally second-order accurate scheme for the displacements (Seidel, 2009). The maximum von Mises stress J_2 was also computed according to

$$J_2 = \frac{1}{6}\left[\left(\sigma_{xx} - \sigma_{yy}\right)^2 + \sigma_{xx}^2 + \sigma_{yy}^2 + 6\sigma_{xy}^2\right].$$

Simulations were run using six systematically-refined mesh levels from a coarse mesh of eight elements to a fine mesh of 8192 elements. Figure 6.4a shows the L_2 norms of the discretization error in the displacements and the discretization error in maximum von Mises stress, with all three quantities displaying convergent behavior. The orders of accuracy for these three quantities are given in Figure 6.4b and show that the displacements asymptote to second-order accuracy while the maximum von Mises stress appears to converge to somewhat less than second order.

6.3 Method of manufactured solutions (MMS)

This section addresses the difficult question of how to create an exact solution for complex PDEs, where complexity refers to characteristics such as nonlinearity, nonconstant coefficients, irregular domain shape, higher dimensions, multiple submodels, and coupled systems of equations. The traditional methods for obtaining exact solutions discussed

earlier generally cannot handle this level of complexity. The primary need for exact solutions to complex PDEs is for order of accuracy testing during code verification.

The *method of manufactured solutions* (MMS) is a general and very powerful approach for creating exact solutions. Rather than trying to find an exact solution to a PDE with given initial and boundary conditions, the goal is to "manufacture" an exact solution to a slightly modified equation. For code verification purposes, it is not required that the manufactured solution be related to a physically realistic problem; recall that code verification deals only with the mathematics of a given problem. The general concept behind MMS is to choose a solution *a priori*, then operate the governing PDEs onto the chosen solution, thereby generating additional analytic source terms that require no discretization. The chosen (manufactured) solution is then the exact solution to the modified governing equations made up of the original equations plus the additional analytic source terms. Thus, MMS involves the solution to the backward problem: given an original set of equations and a chosen solution, find a modified set of equations that the chosen solution will satisfy.

While the MMS approach for generating exact solutions was not new (e.g., see Zadunaisky, 1976; Stetter, 1978), Roache and Steinberg (1984) and Steinberg and Roache (1985) appear to be the first to employ these exact solutions for the purposes of code verification. Their original work looked at the asymptotic rate of convergence of the discretization errors with systematic mesh refinement. Shih (1985) independently developed a similar procedure for debugging scientific computing codes, but without employing mesh refinement to assess convergence or order of accuracy.

The concepts behind MMS for the purpose of code verification were later refined by Roache *et al*. (1990) and Roache (1998). The term "manufactured solution" was coined by Oberkampf *et al*. (1995) and refers to the fact that the method generates (or manufactures) a related set of governing equations for a chosen analytic solution. An extensive discussion of manufactured solutions for code verification is presented by Knupp and Salari (2003) and includes details of the method as well as application to a variety of different PDEs. Recent reviews of the MMS procedure are presented by Roache (2002) and Roy (2005). While it is not uncommon for MMS to be used for verifying computational fluid dynamics codes (e.g., Roache *et al*., 1990; Pelletier *et al*., 2004; Roy *et al*., 2004; Eca *et al*., 2007), it has also begun to appear in other disciplines, for example, fluid–structure interaction (Tremblay *et al*., 2006).

MMS allows the generation of exact solutions to PDEs of nearly arbitrary complexity, with notable exceptions discussed in Section 6.3.3. When combined with the order verification procedure described in Chapter 5, MMS provides a powerful tool for code verification. When physically realistic manufactured solutions are desired, the modified MMS approaches discussed in Section 6.4 can be used.

6.3.1 Procedure

The procedure for creating an exact solution using MMS is fairly straightforward. For a scalar mathematical model, this procedure can be summarized as follows.

Step 1 Establish the mathematical model in the form $L(u) = 0$, where $L(\bullet)$ is the differential operator and u the dependent variable.

Step 2 Choose the analytic form of the manufactured solution $\hat{u}$.

Step 3 Operate the mathematical model $L(\bullet)$ onto the manufactured solution $\hat{u}$. to obtain the analytic source term $s = L(\hat{u})$.

Step 4 Obtain the modified form of the mathematical model by including the analytic source term $L(u) = s$.

Because of the manner in which the analytic source term is obtained, it is a function of the independent variables only and does not depend on u. The initial and boundary conditions can be obtained directly from the manufactured solution $\hat{u}$. For a system of equations, the manufactured solution $\hat{u}$ and the source term s are simply considered to be vectors, but otherwise the process is unchanged.

An advantage of using MMS to generate exact solutions to general mathematical models is that the procedure is not affected by nonlinearities or coupled sets of equations. However, the approach is conceptually different from the standard training scientists and engineers receive in problem solving. Thus it is helpful to examine a simple example which highlights the subtle nature of MMS.

Consider again the unsteady 1-D heat conduction equation that was examined in Section 6.2.1.1. The governing partial differential equation written in the form $L(T) = 0$ is

$$\frac{\partial T}{\partial t} - \alpha \frac{\partial^2 T}{\partial x^2} = 0. \tag{6.19}$$

Once the governing equation has been specified, the next step is to choose the analytic manufactured solution. A detailed discussion on how to choose the solution will follow in Section 6.3.1.1, but for now, consider a combination of exponential and sinusoidal functions:

$$\hat{T}(x, t) = T_o \exp(t/t_o) \sin(\pi x/L). \tag{6.20}$$

Due to the analytic nature of this chosen solution, the derivatives that appear in the governing equation can be evaluated exactly as

$$\frac{\partial \hat{T}}{\partial t} = T_o \sin(\pi x/L) \frac{1}{t_o} \exp(t/t_o)$$
$$\frac{\partial^2 \hat{T}}{\partial x^2} = -T_o \exp(t/t_o) (\pi/L)^2 \sin(\pi x/L).$$

We now modify the mathematical model by including the above derivatives, along with the thermal diffusivity α, on the right hand side of the equation. The modified governing equation that results is

$$\frac{\partial T}{\partial t} - \alpha \frac{\partial^2 T}{\partial x^2} = \left[\frac{1}{t_0} + \alpha \left(\frac{\pi}{L} \right)^2 \right] T_o \exp(t/t_o) \sin(\pi x/L). \tag{6.21}$$

The left hand side of Eq. (6.21) is the same as in the original mathematical model given by Eq. (6.19), thus no modifications to the underlying numerical discretization in the code

under consideration need to be made. The right hand side could be thought of in physical terms as a distributed source term, but in fact it is simply a convenient mathematical construct that will allow straightforward code verification testing. The key concept behind MMS is that the exact solution to Eq. (6.21) is known and is given by Eq. (6.20), the manufactured solution $\hat{T}(x, t)$ that was chosen in the beginning.

As the governing equations become more complex, symbolic manipulation tools such as Mathematica™, Maple™, or MuPAD should be used. These tools have matured greatly over the last two decades and can produce rapid symbolic differentiation and simplification of expressions. Most symbolic manipulation software packages have built-in capabilities to output the solution and the source terms directly as computer source code in both Fortran and C/C^{++} programming languages.

6.3.1.1 Manufactured solution guidelines for code verification

When used for code verification studies, manufactured solutions should be chosen to be analytic functions with smooth derivatives. It is important to ensure that no derivatives vanish, including cross-derivatives if these show up in the governing equations. Trigonometric and exponential functions are recommended since they are smooth and infinitely differentiable. Recall that the order verification procedures involve systematically refining the spatial mesh and/or time step, thus obtaining numerical solutions can be expensive for complex 3-D applications. In some cases, the high cost of performing order verification studies in multiple dimensions using MMS can be significantly reduced simply by reducing the frequency content of the manufactured solution over the selected domain. In other words, it is not important to have a full period of a sinusoidal manufactured solution in the domain, often only a fraction (one-third, one-fifth, etc.) of a period is sufficient to exercise the terms in the code.

Although the manufactured solutions do not need to be physically realistic when used for code verification, they should be chosen to obey certain physical constraints. For example, if the code requires the temperature to be positive (e.g., in the evaluation of the speed of sound which involves the square root of the temperature), then the manufactured solution should be chosen to give temperature values significantly larger than zero.

Care should be taken that one term in the governing equations does not dominate the other terms. For example, even if the actual application area for a Navier–Stokes code will be for high-Reynolds number flows, when performing code verification studies, the manufactured solution should be chosen to give Reynolds numbers near unity so that convective and diffusive terms are of the same order of magnitude. For terms that have small relative magnitudes (e.g., if the term is scaled by a small parameter such as 1/Re), mistakes can still be found through order verification, but possibly only on extremely fine meshes.

A more rigorous approach to ensuring that all of the terms in the governing equations are roughly the same order of magnitude over some significant region of the domain is to examine ratios of those terms. This process has been used by Roy *et al*. (2007b) as part of the order of accuracy verification of a computational fluid dynamics code including

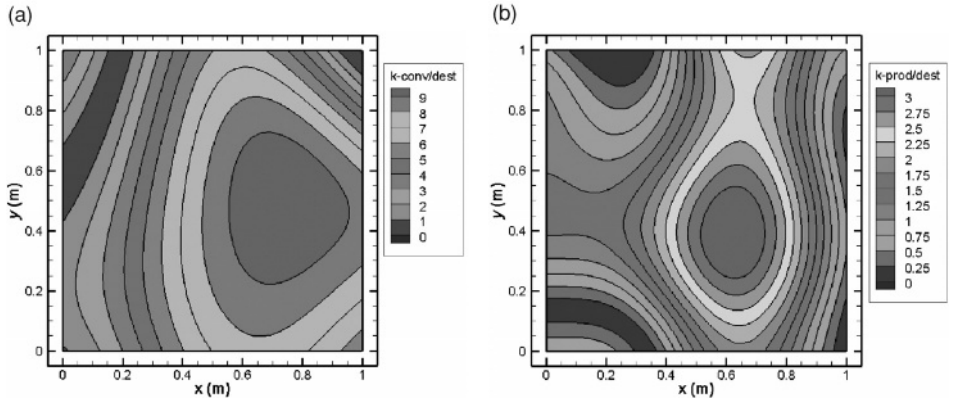

Figure 6.5 Ratios of (a) the convection terms and (b) the production terms to the destruction terms for a turbulent kinetic energy transport equation (from Roy *et al*., 2007b).

a complicated two-equation turbulence model. Example ratios for different terms in the equation governing the transport of turbulent kinetic energy are given in Figure 6.5. These plots show that the convection, production, and destruction terms in this transport equation are of roughly the same order of magnitude over most of the domain.

6.3.1.2 Boundary and initial conditions

The discretization schemes for the PDE and the various submodels comprise a significant fraction of the possible options in most scientific computing codes. When performing code verification studies on these options, there are two approaches for handling boundary and initial conditions. The first approach is to impose the mathematically consistent boundary and initial conditions that are required according to the mathematical character of the differential equations. For example, if Dirichlet (fixed-value) or Neumann (fixed-gradient) boundary conditions are required, these can be determined directly from the analytic manufactured solution (although these will generally not be constant along the boundary). The second option is to simply specify all boundary values with Dirichlet or Neumann values from the manufactured solution. This latter approach, although mathematically ill-posed, often does not adversely affect the order of accuracy test. In any case, over-specification of boundary conditions will not lead to a false positive for order of accuracy testing (i.e., a case where the order of accuracy is verified but there is a mistake in the code or inconsistency in the discrete algorithm).

In order to verify the implementation of the boundary conditions, the manufactured solution should be tailored to exactly satisfy a given boundary condition on a domain boundary. Bond *et al*. (2007) present a general approach for tailoring manufactured solutions to ensure that a given boundary condition is satisfied along a general boundary. The method involves multiplying any standard manufactured solution by a function which has values and/or derivatives which are zero over a specified boundary. To modify the standard form of the manufactured solution for a 2-D steady-state solution for temperature, one may simply

write the manufactured solution as follows:

$$\hat{T}(x, y) = T_0 + \hat{T}_1(x, y), \tag{6.22}$$

where $\hat{T}_1(x, y)$ is any baseline manufactured solution. For example, this manufactured solution could take the form

$$\hat{T}_1(x, y) = T_x f_s\left(\frac{a_x \pi x}{L}\right) + T_y f_s\left(\frac{a_y \pi y}{L}\right), \tag{6.23}$$

where the $f_s(\cdot)$ functions represent a mixture of sines and cosines and T_x, T_y, a_x, and a_y are constants (note the subscripts used here do not denote differentiation).

For 2-D problems, a boundary can be represented by the general curve $F(x, y) = C$, where C is a constant. A new manufactured solution appropriate for verifying boundary conditions can be found by multiplying the spatially-varying portion of $\hat{T}(x, y)$ by the function $[C - F(x, y)]^m$, i.e.,

$$\hat{T}_{\mathrm{BC}}(x, y) = T_0 + \hat{T}_1(x, y)\,[C - F(x, y)]^m. \tag{6.24}$$

This procedure will ensure that the manufactured solution is equal to the constant T_0 along the specified boundary for $m = 1$. Setting $m = 2$, in addition to enforcing the above Dirichlet BC for temperature, will ensure that the gradient of temperature normal to the boundary is equal to zero, i.e., the adiabatic boundary condition for this 2-D steady heat conduction example. In practice, the curve $F(x, y) = C$ is used to define the domain boundary where the boundary conditions will be tested.

To illustrate this procedure, we will choose the following simple manufactured solution for temperature:

$$\hat{T}(x, y) = 300 + 25\cos\left(\frac{7}{4}\frac{\pi x}{L}\right) + 40\sin\left(\frac{4}{3}\frac{\pi y}{L}\right), \tag{6.25}$$

where the temperature is assumed to be in units of Kelvin. For the surface where the Dirichlet or Neumann boundary condition will be applied, choose:

$$F(x, y) = \frac{1}{2}\cos\left(\frac{0.4\pi x}{L}\right) - \frac{y}{L} = 0. \tag{6.26}$$

A mesh bounded on the left by $x/L = 0$, on the right by $x/L = 1$, on the top by $y/L = 1$, and on the bottom by the above defined surface $F(x, y) = 0$ is shown in Figure 6.6a. The standard manufactured solution given by Eq. (6.25) is also shown in Figure 6.6b, where clearly the constant value and gradient boundary conditions are not satisfied.

Combining the baseline manufactured solution with the boundary specification yields

$$\hat{T}_{\mathrm{BC}}(x, y) = 300 + \left[25\cos\left(\frac{7}{4}\frac{\pi x}{L}\right) + 40\sin\left(\frac{4}{3}\frac{\pi y}{L}\right)\right]\left[-\frac{1}{2}\cos\left(\frac{0.4\pi x}{L}\right) + \frac{y}{L}\right]^m. \tag{6.27}$$

When $m = 1$, the curved lower boundary will satisfy the constant temperature condition of 300 K as shown in the manufactured solutions of Figure 6.7a. When $m = 2$ (Figure 6.7b),

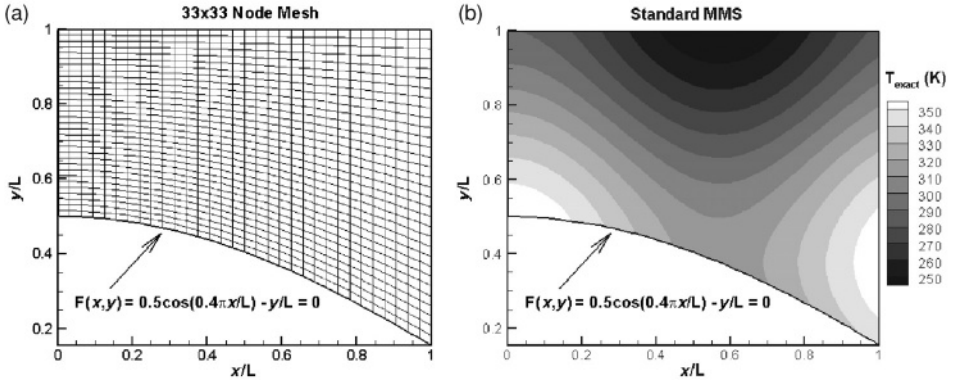

Figure 6.6 Grid (a) and baseline manufactured solution (b) for the MMS boundary condition example.

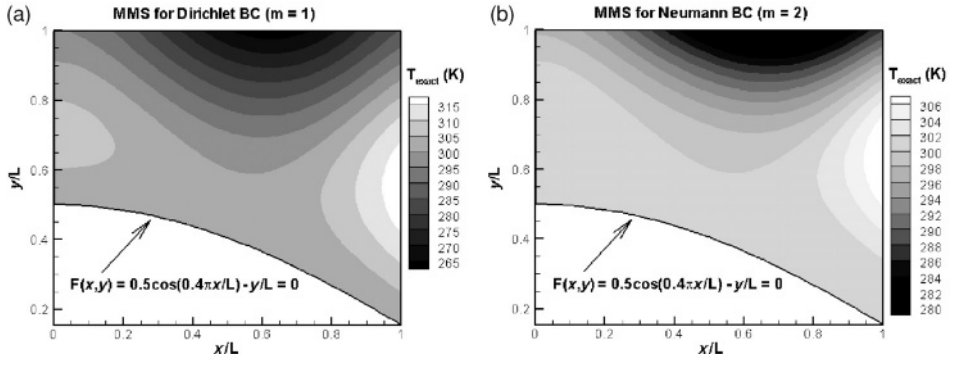

Figure 6.7 Manufactured solutions given by Eq. (6.27): (a) fixed temperature boundary condition ($m = 1$) and (b) fixed temperature and zero gradient (adiabatic) boundary condition ($m = 2$).

the zero gradient (i.e., adiabatic) boundary condition is also satisfied. For more details on this approach as well as extensions to 3-D, see Bond *et al*. (2007).

6.3.2 Benefits of MMS for code verification

There are a number of benefits to using the MMS procedure for code verification. Perhaps the most important benefit is that it handles complex mathematical models without additional difficulty since the procedures described above are readily extendible to nonlinear, coupled systems of equations. In addition, MMS can be used to verify most of the coding options available in scientific computing codes, including the consistency/convergence of numerical algorithms. The procedure is not tied to a specific discretization scheme, but works equally well for finite-difference, finite-volume, and finite-element methods.

The use of MMS for code verification has been shown to be remarkably sensitive to mistakes in the discretization (see for example Chapter 3.11 of Roache (1998)). In one

particular case of a compressible Navier–Stokes code for unstructured grids (Roy *et al*., 2007b), global norms of the discretization error were found to be non-convergent. Further investigation found a small discrepancy (in the fourth significant figure!) for the constant thermal conductivity between the governing equations used to generate the source terms and the model implementation in the code. When this discrepancy was corrected, the order verification test was passed. The same study uncovered an algorithm inconsistency in the discrete formulation of the diffusion operator that resulted in non-ordered behavior on skewed meshes. This same formulation had been implemented in at least one commercial computational fluid dynamics code (see Roy *et al*. (2007b) for more details).

In addition to its ability to indicate the presence of coding mistakes (i.e., bugs), the MMS procedure combined with order verification is also a powerful tool for finding and removing those mistakes (i.e., debugging). After a failed order of accuracy test, individual terms in the mathematical model and the numerical discretization can be omitted, allowing the user to quickly isolate the terms with the coding mistake. When combined with the approach for verifying boundary conditions discussed in the previous section and a suite of meshes with different topologies (e.g., hexahedral, prismatic, tetrahedral, and hybrid meshes, as discussed in Chapter 5), the user has a comprehensive set of tools to aid in code debugging.

6.3.3 Limitations of MMS for code verification

There are some limitations to using the MMS procedure for code verification. The principal disadvantage is that it requires the user to incorporate arbitrary source terms, initial conditions, and boundary conditions in a code. Even when the code provides a framework for including these additional interfaces, their specific form changes for each different manufactured solution. The MMS procedure is thus code-intrusive and generally cannot be performed as a black-box testing procedure where the code simply returns some outputs based on a given set of inputs. In addition, each code option which changes the mathematical model requires new source terms to be generated. Thus order verification with MMS can be time consuming when many code options must be verified.

Since the MMS procedure for code verification relies on having smooth solutions, the analysis of discontinuous weak solutions (e.g., solutions with shock-waves) is still an open research issue. Some traditional exact solutions exist for discontinuous problems such as the generalized Riemann problem (Gottlieb and Groth, 1988) and more complicated solutions involving shock waves and detonations have been developed that involve infinite series (Powers and Stewart, 1992) or a change of dependent variable (Powers and Aslam, 2006). However, to our knowledge, discontinuous manufactured solutions have not been created. Such "weak" exact solutions are needed for verifying codes used to solve problems with discontinuities.

Difficulties also arise when applying MMS to mathematical models where the governing equations themselves contain `min`, `max`, or other nonsmooth switching functions. These functions generally do not result in continuous manufactured solution source terms. These

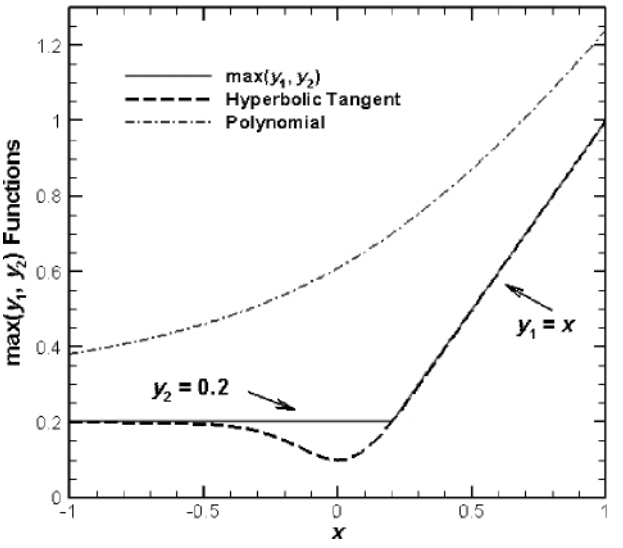

Figure 6.8 Examples of smooth approximations for $\max(y_1, y_2)$ using the hyperbolic tangent from Eq. (6.28) and the polynomial from Eq. (6.29).

switching functions can be dealt with by simply turning off different branches of the switching functions (Eca *et al.*, 2007) or by tailoring the manufactured solution so that only one switching branch will be used for a given verification test (Roy *et al.*, 2007b). The former is simpler but more code intrusive than the latter approach. We recommend that model developers employ smooth blending functions such as the hyperbolic tangent both to simplify the code verification testing and to possibly make the numerical solution process more robust. For example, consider the function $\max(y_1, y_2)$, where y_1 and y_2 are given by

$$y_1(x) = x,$$
$$y_2 = 0.2.$$

One approach for smoothing this max function in the region of $x = 0.2$ is the hyperbolic tangent smoothing function given by

$$\max(y_1, y_2) \approx F y_1 + (1 - F) y_2,$$
$$\text{where } F = \frac{1}{2}\left[\tanh\left(y_1 / y_2\right) + 1\right]. \tag{6.28}$$

Another approach is to use the following polynomial expression:

$$\max(y_1, y_2) \approx \frac{\sqrt{(y_1 - y_2)^2 + 1} + y_1 + y_2}{2}. \tag{6.29}$$

The two approximations of $\max(y_1, y_2)$ are shown graphically in Figure 6.8. The hyperbolic tangent approximation provides less error relative to the original $\max(y_1, y_2)$ function, but

creates an inflection point where the first derivative (slope) of this function will change sign. The polynomial function is monotone, but gives a larger error magnitude. Models that rely on tabulated data (i.e., look-up tables) suffer from similar problems, and smooth approximations of such data should be considered. MMS is also limited when the mathematical model contains complex algebraic submodels which do not have closed-form solutions and thus must be solved numerically (e.g., by a root-finding algorithm). Such complex submodels are best addressed separately through unit and/or component level software testing discussed in Chapter 4.

6.3.4 Examples of MMS with order verification

Two examples are now presented which use MMS to generate exact solutions. These manufactured solutions are then employed for code verification using the order of accuracy test.

6.3.4.1 2-D steady heat conduction

Order verification using MMS has been applied to steady-state heat conduction with a constant thermal diffusivity. The governing equation simply reduces to Poisson's equation for temperature:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = s(x, y), \tag{6.30}$$

where $s(x, y)$ is the manufactured solution source term. Coordinate transformations of the form

$$(x, y) \rightarrow (\xi, \eta)$$

are used to globally transform the governing equation into a Cartesian computational space with $\Delta\xi = \Delta\eta = 1$ (Thompson *et al.*, 1985). The transformed governing equation thus becomes

$$\frac{\partial F_1}{\partial \xi} + \frac{\partial G_1}{\partial \eta} = \frac{s(x, y)}{J}, \tag{6.31}$$

where J is the Jacobian of the mesh transformation. The fluxes F_1 and G_1 are defined as

$$F_1 = \frac{\xi_x}{J} F + \frac{\xi_y}{J} G,$$
$$G_1 = \frac{\eta_x}{J} F + \frac{\eta_y}{J} G,$$

where

$$F = \xi_x \frac{\partial T}{\partial \xi} + \eta_x \frac{\partial T}{\partial \eta},$$
$$G = \xi_y \frac{\partial T}{\partial \xi} + \eta_y \frac{\partial T}{\partial \eta}.$$

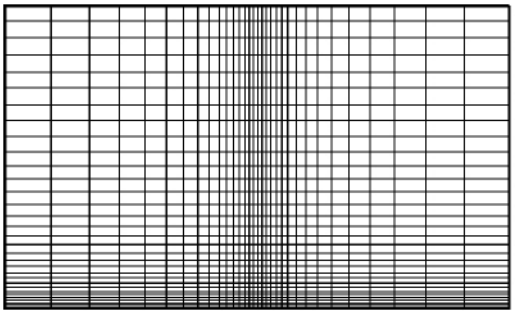

Figure 6.9 Stretched Cartesian mesh with 33×33 nodes for the 2-D steady heat conduction problem.

An explicit point Jacobi method (Tannehill *et al*., 1997) is used to advance the discrete equations towards the steady-state solution. Standard, three-point, centered finite differences are used in the transformed coordinates and central differences are also employed for the grid transformation metric terms (x_ξ, x_η, y_ξ, etc.), thus resulting in a discretization scheme that is formally second-order accurate in space. The numerical solutions were iteratively converged to machine zero, i.e., the iterative residuals were reduced by approximately 14 orders of magnitude for the double precision computations employed. Thus iterative and round-off error are assumed to be negligible and the numerical solutions are effectively the exact solution to the discrete equation.

The following manufactured solution was chosen

$$\hat{T}(x, y) = T_0 + T_x \cos\left(\frac{a_x \pi x}{L}\right) + T_y \sin\left(\frac{a_y \pi y}{L}\right) + T_{xy} \sin\left(\frac{a_{xy} \pi x y}{L^2}\right), \qquad (6.32)$$

where

$$T_0 = 400 \text{ K}, \quad T_x = 45 \text{ K}, \quad T_y = 35 \text{ K}, \quad T_{xy} = 27.5 \text{ K},$$
$$a_x = 1/3, \quad a_y = 1/4, \quad a_{xy} = 1/2, \quad L = 5 \text{ m},$$

and Dirichlet (fixed-value) boundary conditions were applied on all four boundaries as determined by Eq. (6.32). A family of stretched Cartesian meshes was created by first generating the finest mesh (129×129 nodes), and then successively eliminating every other gridline to create the coarser meshes. Thus systematic refinement (or coarsening in this case) is ensured. The 33×33 node mesh is presented in Figure 6.9, showing significant stretching in the x-direction in the center of the domain and in the y-direction near the bottom boundary. The manufactured solution from Eq. (6.32) is shown graphically in Figure 6.10. The temperature varies smoothly over the domain and the manufactured solution gives variations in both coordinate directions.

Discrete L_2 norms of the discretization error (i.e., the difference between the numerical solution and the manufactured solution) were computed for grid levels from 129×129 nodes ($h = 1$) to 9×9 nodes ($h = 16$). These norms are given in Figure 6.11a and closely follow the expected second order slope. The observed order of accuracy of the L_2 norms of the discretization error was computed from Eq. (5.23) for successive mesh levels, and the

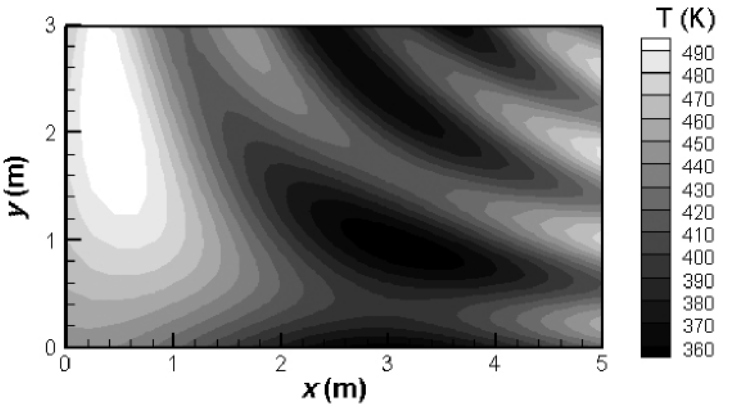

Figure 6.10 Manufactured solution for temperature for the 2-D steady heat conduction problem.

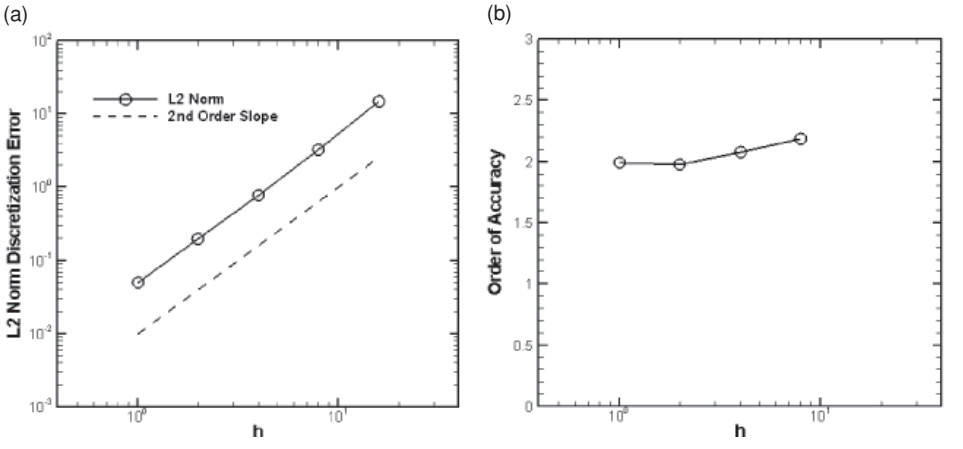

Figure 6.11 Code verification for the 2-D steady heat conduction problem: (a) discrete L2 norms of the discretization error and (b) observed order of accuracy.

results are shown in Figure 6.11b. The observed order of accuracy is shown to converge to the formal order of two as the meshes are refined, thus the code is considered to be verified for the options tested. Note that while the discrete transformations were tested with respect to the clustering of the mesh, this choice of grid topologies would not test the implementation of the grid metric terms dealing with cell skewness or coordinate rotation (e.g., ξ_y, η_x).

6.3.4.2 2D Steady Euler equations

This MMS example deals with the Euler equations, which govern the flow of an inviscid fluid. This example is adapted from Roy *et al.* (2004) and we will demonstrate the steps of both generating the exact solution with MMS and the order verification procedure. The

two-dimensional, steady-state form of the Euler equations is given by

$$\begin{aligned}
\frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} &= s_m(x, y), \\
\frac{\partial(\rho u^2 + p)}{\partial x} + \frac{\partial(\rho uv)}{\partial y} &= s_x(x, y), \\
\frac{\partial(\rho vu)}{\partial x} + \frac{\partial(\rho v^2 + p)}{\partial y} &= s_y(x, y), \\
\frac{\partial(\rho u e_t + pu)}{\partial x} + \frac{\partial(\rho v e_t + pv)}{\partial y} &= s_e(x, y),
\end{aligned} \tag{6.33}$$

where arbitrary source terms $s(x, y)$ are included on the right hand side for use with MMS. In these equations, u and v are the Cartesian components of velocity in the x- and y-directions, ρ the density, p the pressure, and e_t is the specific total energy, which for a calorically perfect gas is given by

$$e_t = \frac{1}{\gamma - 1} RT + \frac{u^2 + v^2}{2}, \tag{6.34}$$

where R is the specific gas constant, T the temperature, and γ the ratio of specific heats. The final relation used to close the set of equations is the perfect gas equation of state:

$$p = \rho RT. \tag{6.35}$$

The manufactured solutions for this case are chosen as simple sinusoidal functions given by

$$\begin{aligned}
\rho(x, y) &= \rho_0 + \rho_x \sin\left(\frac{a_{\rho x}\pi x}{L}\right) + \rho_y \cos\left(\frac{a_{\rho y}\pi y}{L}\right), \\
u(x, y) &= u_0 + u_x \sin\left(\frac{a_{ux}\pi x}{L}\right) + u_y \cos\left(\frac{a_{uy}\pi y}{L}\right), \\
v(x, y) &= v_0 + v_x \cos\left(\frac{a_{vx}\pi x}{L}\right) + v_y \sin\left(\frac{a_{vy}\pi y}{L}\right), \\
p(x, y) &= p_0 + p_x \cos\left(\frac{a_{px}\pi x}{L}\right) + p_y \sin\left(\frac{a_{py}\pi y}{L}\right).
\end{aligned} \tag{6.36}$$

The subscripts here refer to constants (not differentiation) with the same units as the variable, and the dimensionless constants a generally vary between 0.5 and 1.5 to provide low frequency solutions over an $L \times L$ square domain. For this example, the constants were chosen to give supersonic flow in both the positive x and positive y directions. While not necessary, this choice simplifies the inflow boundary conditions to Dirichlet (specified) values at the inflow boundaries, whereas outflow boundary values are simply extrapolated from the interior. The inflow boundary conditions are specified directly from the manufactured solution. The specific constants chosen for this example are shown in Table 6.2, and a plot of the manufactured solution for the density is given in Figure 6.12. The density varies smoothly in both coordinate directions between 0.92 and 1.13 kg/m^3.

Table 6.2 *Constants for the supersonic Euler manufactured solution*

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

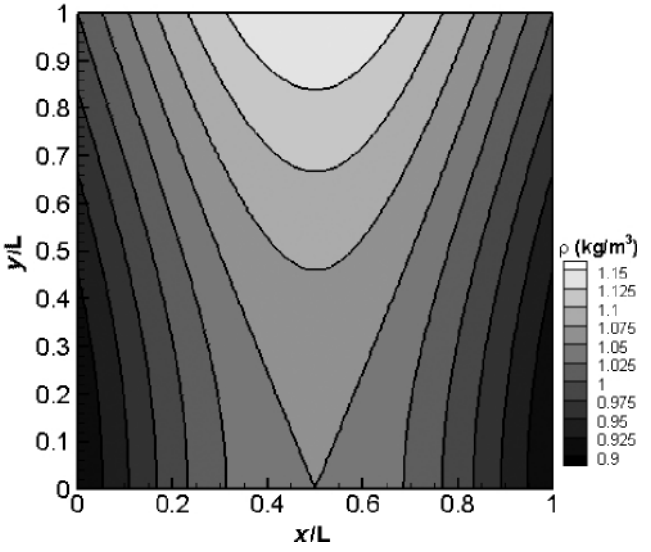

Figure 6.12 Manufactured solution for density for the Euler equations.

Substitution of the chosen manufactured solutions into the governing equations allows the analytic determination of the source terms. For example, the source term for the mass conservation equation is given by:

$$
\begin{aligned}
s_{\mathrm{m}}(x, y) = & \frac{a_{ux}\pi u_x}{L} \cos\left(\frac{a_{ux}\pi x}{L}\right)\left[\rho_0 + \rho_x \sin\left(\frac{a_{\rho x}\pi x}{L}\right) + \rho_y \cos\left(\frac{a_{\rho y}\pi y}{L}\right)\right] \\
& + \frac{a_{vy}\pi v_y}{L} \cos\left(\frac{a_{vy}\pi y}{L}\right)\left[\rho_0 + \rho_x \sin\left(\frac{a_{\rho x}\pi x}{L}\right) + \rho_y \cos\left(\frac{a_{\rho y}\pi y}{L}\right)\right] \\
& + \frac{a_{\rho x}\pi \rho_x}{L} \cos\left(\frac{a_{\rho x}\pi x}{L}\right)\left[u_0 + u_x \sin\left(\frac{a_{ux}\pi x}{L}\right) + u_y \cos\left(\frac{a_{uy}\pi y}{L}\right)\right] \\
& + \frac{a_{\rho y}\pi \rho_y}{L} \sin\left(\frac{a_{\rho y}\pi y}{L}\right)\left[v_0 + v_x \cos\left(\frac{a_{vx}\pi x}{L}\right) + v_y \sin\left(\frac{a_{vy}\pi y}{L}\right)\right].
\end{aligned}
$$

The source terms for the momentum and energy equations are significantly more complex, and all source terms were obtained using Mathematica™. A plot of the source term for the energy conservation equation is given in Figure 6.13. Note the smooth variations of the source term in both coordinate directions.

Table 6.3 *Cartesian meshes employed in the Euler manufactured solution*

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

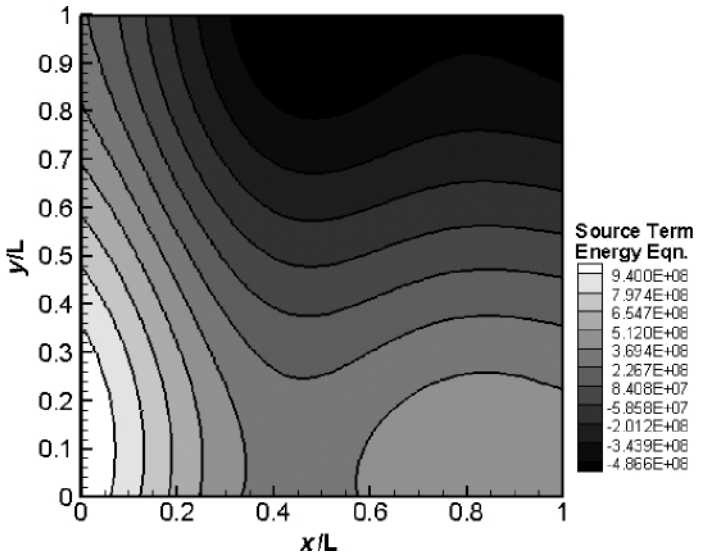

Figure 6.13 Analytic source term for the energy conservation equation.

The governing equations are discretized and solved on multiple meshes. In this case, two different finite-volume computational fluid dynamics codes were employed: Premo, an unstructured grid code, and Wind, a structured grid code (see Roy *et al*. (2004) for more details). Both codes utilized the second-order Roe upwind scheme for the convective terms (Roe, 1981). The formal order of accuracy of both codes is thus second order for smooth problems.

The five Cartesian meshes employed are summarized in Table 6.3. The coarser meshes were found by successively eliminating every other grid line from the fine mesh (i.e., a grid refinement factor of $r = 2$). It is important to note that while the current example was performed on Cartesian meshes for simplicity, a more general code verification analysis would employ the most general meshes which will be used by the code (e.g., unstructured

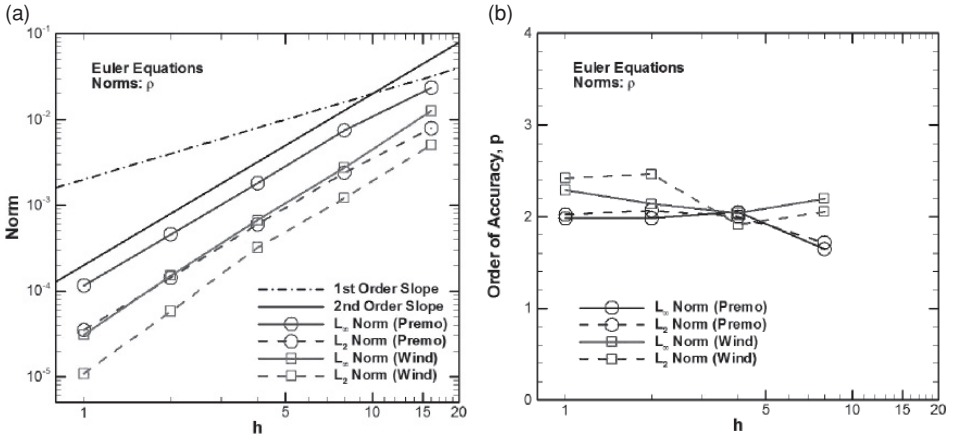

Figure 6.14 Code verification results for the 2-D steady Euler equations: (a) norms of the discretization error and (b) observed order of accuracy.

meshes with significant stretching, skewness, boundary orientations). See Section 5.4 for additional discussion of mesh topology issues.

The global discretization error is measured using discrete L_∞ and L_2 norms of the discretization error, where the exact solution comes directly from the chosen manufactured solution given by Eqs. (6.36). The behavior of these two discretization error norms for the density ρ as a function of the cell size h is given in Figure 6.14a. On the logarithmic scale, a first-order scheme will display a slope of unity, while a second-order scheme will give a slope of two. The discretization error norms for the density appear to converge with second-order accuracy.

A more quantitative method for assessing the observed order of accuracy is to calculate it using the norms of the discretization error. Since the exact solution is known, the relation for the observed order of accuracy of the discretization error norms comes from Eq. (5.23). A plot of the observed order of accuracy as a function of the element size h is presented in Figure 6.14b. The Premo code clearly asymptotes to second-order accuracy, while the Wind code appears to asymptote to an order of accuracy that is slightly higher than two. In general, an observed order of accuracy higher than the formal order can occur due to error cancellation and should not be considered as a failure of the order verification test (although it may indicate mistakes is determining the formal order of accuracy of the method). Further grid refinement would possibly provide more definitive results for the Wind code. In this case, the observed order of accuracy for both codes is near two, thus the formal order of accuracy is recovered, and the two codes are considered verified for the options examined.

6.4 Physically realistic manufactured solutions

The MMS procedure discussed in the previous section is the most general method for obtaining exact solutions to mathematical models for use in code verification studies. Since

physical realism of the solutions is not required during code verification, the solutions are somewhat arbitrary and can be tailored to exercise all terms in the mathematical model. However, there are many cases in which physically realistic exact solutions are desired, such as assessing sensitivity of a numerical scheme to mesh quality, evaluating the reliability of discretization error estimators, and judging the overall effectiveness of solution adaptation schemes. There are two main approaches for obtaining physically realistic manufactured solutions to complex equations, and these two approached are discussed below.

6.4.1 Theory-based solutions

One approach to obtaining physically realistic manufactured solutions is to use a simplified theoretical model of the physical phenomenon as a basis for the manufactured solution. For example, if a physical process is known to exhibit an exponential decay in the solution with time, then a manufactured solution of the form

$$\alpha \exp(-\beta t)$$

could be employed, where α and β could be chosen to provide physically meaningful solutions.

There are two examples of this approach applied to the modeling of fluid turbulence. Pelletier *et al*. (2004) have verified a 2-D incompressible finite element code that employs a k-ε two-equation turbulence model. They constructed manufactured solutions which mimic turbulent shear flows, with the turbulent kinetic energy and the turbulent eddy viscosity as the two quantities specified in the manufactured solution. More recently, Eca and Hoekstra (2006) and Eca *et al*. (2007) developed physically realistic manufactured solutions mimicking steady, wall-bounded turbulence for 2-D incompressible Navier–Stokes codes. They examined both one- and two-equation turbulence models and noted challenges in generating physically realistic solutions in the near-wall region.

6.4.2 Method of nearby problems (MNP)

The second approach for generating physically realistic manufactured solutions is called the *method of nearby problems* (MNP) and was proposed by Roy and Hopkins (2003). This approach involves first computing a highly-refined numerical solution for the problem of interest, then generating an accurate curve fit of that numerical solution. If both the underlying numerical solution and the curve fit are "sufficiently" accurate, then it will result in a manufactured solution which has small source terms. The sufficiency conditions for the "nearness" of the problem have been explored for first-order quasi-linear ordinary differential equations (Hopkins and Roy, 2004) but rigorous bounds on this nearness requirement for PDEs have not yet been developed.

MNP has been successfully demonstrated for one-dimensional problems by Roy *et al*. (2007a) who used the procedure to create a nearby solution to the steady-state Burgers'

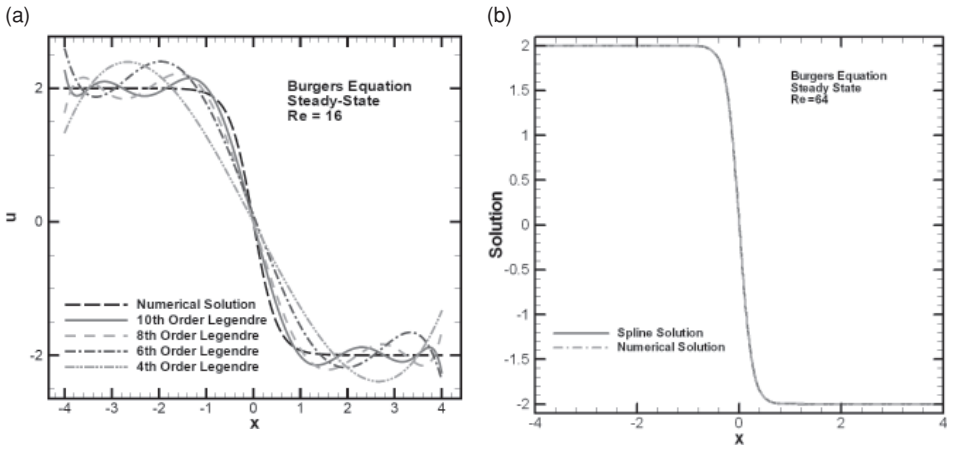

Figure 6.15 Examples of curve fitting for the viscous shock wave solution to Burgers' equation: (a) global Legendre polynomial fits for Re = 16 and (b) fifth-order Hermite splines for Re = 64 (from Roy *et al.*, 2007a).

equation for a viscous shock wave. They used fifth-order Hermite splines to generate the exact solutions for Reynolds numbers of 8, 64, and 512. To explain why spline fits must be used, rather than global curve fits, consider Figure 6.15. Global Legendre polynomial fits for the steady-state Burgers' equation for a viscous shock at a Reynolds number of 16 are given in Figure 6.15a. Not only is the viscous shock wave not adequately resolved, but the global fits also exhibit significant oscillations at the boundaries. Hermite spline fits for an even higher Reynolds number of 64 are given in Figure 6.15b, with the spline fit in very good qualitative agreement with the underlying numerical solution. MNP has been extended to 2-D problems by Roy and Sinclair (2009) and the further extension to higher dimensions is straightforward. A 2-D example of MNP used to generate an exact solution to the incompressible Navier–Stokes equations will be given in Section 6.4.2.2.

6.4.2.1 Procedure

The steps for generating physically realistic manufactured solutions using the MNP approach are:

1 compute the original numerical solution on a highly refined grid,
2 generate an accurate spline or curve fit to this numerical solution, thereby providing an analytic representation of the numerical solution,
3 operate the governing partial differential equations on the curve fit to generate analytic source terms (which ideally will be small), and
4 create the nearby problem by appending the analytic source terms to the original mathematical model.

If the source terms are indeed small, then the new problem will be "near" the original one, hence the name "method of nearby problems." The key point to this approach is that, by

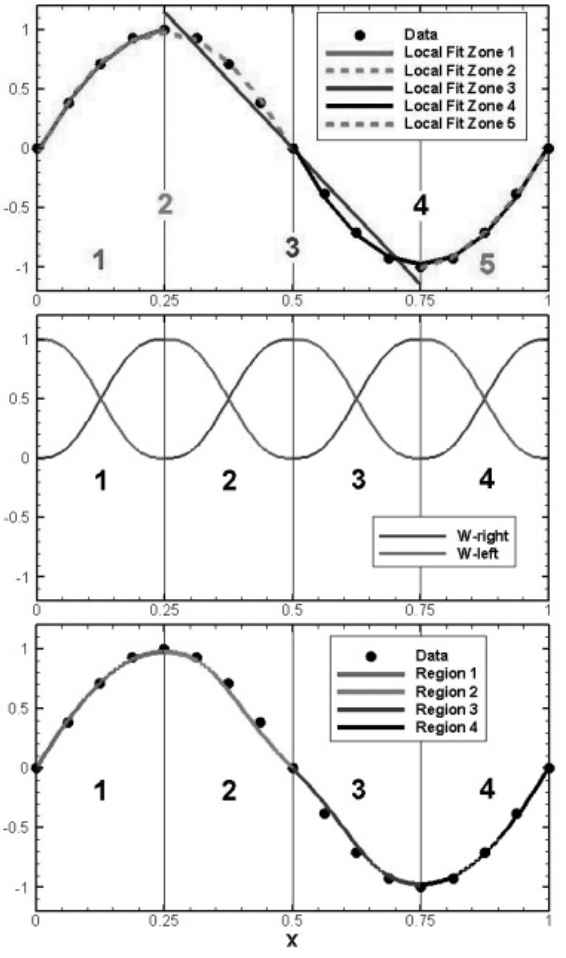

Figure 6.16 Simple one-dimensional example of the weighting function approach for combining local quadratic least squares fits to generate a C^2 continuous spline fit: local fits (top), weighting functions (middle), and resulting C^2 continuous spline fit (bottom) (from Roy and Sinclair, 2009). (See color plate section.)

definition, the curve fit generated in step 2 is the exact solution to the nearby problem. While the approach is very similar to MMS, in MNP the addition of the curve fitting step is designed to provide a physically realistic exact solution.

To demonstrate the MNP procedure, a simple 1-D example is presented in Figure 6.16, where the original data used to generate the curve fit are 17 points sampled at equal intervals from the function $\sin(2\pi x)$. The goal of this example is to create a spline fit made up of four spline regions that exhibits C^2 continuity at the spline zone interfaces (i.e., continuity up to the second derivative). The spline fitting is performed in a manner that allows arbitrary levels of continuity at spline boundaries and is readily extendible to multiple dimensions following Junkins *et al*. (1973).

The first step is to generate five overlapping local fits Z_1 through Z_5, with each of the interior fits spanning two spline regions (see top of Figure 6.16). A least squares method is used to find a best-fit quadratic function in each of the five regions:

$$Z_n(\bar{x}) = a_n + b_n\bar{x} + c_n\bar{x}^2. \tag{6.37}$$

The overbars in Eq. (6.37) specify that the spatial coordinate x is locally transformed to satisfy $0 \le \bar{x} \le 1$ in each of the interior spline zones. Since each spline zone now has two different local fits, one from the left and the other from the right, these two local fits are combined together with the left and right weighting functions shown in Figure 6.16 (middle). The form of the 1-D weighting function used here for C^2 continuity is

$$W_{\text{right}}(\bar{x}) = \bar{x}^3\left(10 - 15\bar{x} + 6\bar{x}^2\right)$$

and the corresponding left weighting function is defined simply as

$$W_{\text{left}}(\bar{x}) = W_{\text{right}}(1 - \bar{x}).$$

Thus the final fit in each region can be written as

$$F(x, y) = W_{\text{left}}Z_{\text{left}} + W_{\text{right}}Z_{\text{right}}.$$

For example, for region 2, one would have $Z_{\text{left}} = Z_2$ and $Z_{\text{right}} = Z_3$. Note that, in addition to providing the desired level of continuity at spline boundaries, the weighting functions are also useful in reducing the dependence near the boundaries of the local fits where they often exhibit the poorest agreement with the original data. When these final fits are plotted (bottom of Figure 6.16), we see that they are indeed C^2 continuous, maintaining continuity of the function value, slope, and curvature at all three interior spline boundaries.

6.4.2.2 *Example exact solution: 2-D steady Navier–Stokes equations*

An example of the use of MNP to generate physically realistic manufactured solutions is now given for the case of viscous, incompressible flow in a lid-driven cavity at a Reynolds number of 100 (Roy and Sinclair, 2009). This flow is governed by the incompressible Navier–Stokes equations, which for constant transport properties are given by

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = s_{\text{m}}(x, y),$$

$$\rho u\frac{\partial u}{\partial x} + \rho v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} - \mu\frac{\partial^2 u}{\partial x^2} - \mu\frac{\partial^2 u}{\partial y^2} = s_x(x, y),$$

$$\rho u\frac{\partial v}{\partial x} + \rho v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} - \mu\frac{\partial^2 v}{\partial x^2} - \mu\frac{\partial^2 v}{\partial y^2} = s_y(x, y),$$

where $s(x, y)$ are the manufactured solution source terms. These equations are solved in finite-difference form on a standard Cartesian mesh by integrating in pseudo-time using Chorin's artificial compressibility method (Chorin, 1967). In addition, second- and fourth-derivative damping (Jameson *et al.*, 1981) was employed to prevent odd–even decoupling (i.e., oscillations) in the solution. Dirichlet boundary conditions are used for velocity, with

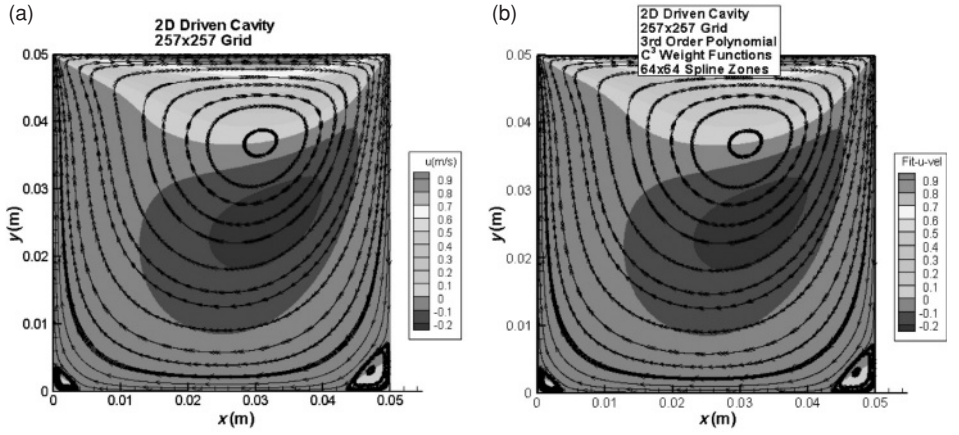

Figure 6.17 Contours of u-velocity and streamlines for a lid-driven cavity at Reynolds number 100: (a) 257×257 node numerical solution and (b) C^3 continuous spline fit using 64×64 spline zones (from Roy and Sinclair, 2009). (See color plate section.)

all boundary velocities equal to zero except for the u-velocity on the top wall which is set to 1 m/s.

A contour plot of the u-velocity (i.e., the velocity in the x-direction) from a numerical solution on a 257×257 grid is given in Figure 6.17a. Also shown in the figure are streamlines which denote the overall clockwise circulation induced by the upper wall velocity (the upper wall moves from left to right) as well as the two counter-clockwise rotating vortices in the bottom corners. A spline fit was generated using third order (i.e., bi-cubic) polynomials in x and y with C^3 continuous weighting functions and 64×64 spline zones. Note that while no additional boundary constraints are placed on the velocity components for the spline fit, the maximum deviations from the original boundary conditions are on the order of 1×10^{-7} m/s and are thus quite small. The u-velocity contours and streamlines for the spline fit are presented in Figure 6.17b. The fit solution is qualitatively the same as the underlying numerical solution. The streamlines were injected at exactly the same locations in both figures and are indistinguishable from each other. Furthermore, in both cases the streamlines near the center of the cavity follow the same path for multiple revolutions.

A more quantitative comparison between the underlying numerical solution and the spline fits is presented in Figure 6.18, which shows discrete norms of the spline fitting error in u-velocity relative to the underlying numerical solution as a function of the number of spline zones in each direction. The average error magnitude (L_1 norm) decreases from 1×10^{-3} m/s to 3×10^{-6} m/s with increasing number of spline zones from 8×8 to 64×64, while the maximum error (infinity norm) decreases from 0.7 m/s to 0.01 m/s.

6.5 Approximate solution methods

This section describes three methods for approximating exact solutions to mathematical models. The first two, series and similarity solutions, are often considered to be exact, but are

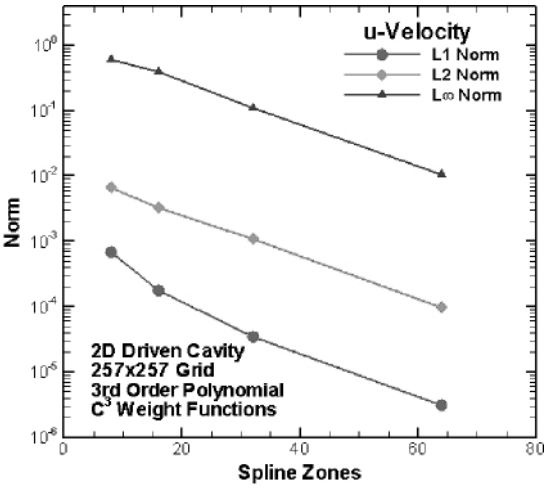

Figure 6.18 Variation of the error in u-velocity between the spline fits and the underlying 257×257 numerical solution as a function of the number of spline zones in each direction for the lid-driven cavity at Reynolds number 100 (from Roy and Sinclair, 2009).

treated as approximate here since we assume that numerical values for the solution must be computed. Furthermore, infinite series and similarity solutions are usually only available for simple PDEs. The third method involves computing a highly-accurate numerical solution to a given problem and is called a numerical benchmark solution.

6.5.1 Infinite series solutions

Solutions involving infinite series are sometimes used to solve differential equations with general boundary and initial conditions. The primary application of series solutions has been for linear differential equations, but they are also a useful tool for obtaining solutions to certain nonlinear differential equations. While these solutions are "analytic," they are not closed form solutions since they involve infinite series. When using infinite series as an approximation of an exact solution to the mathematical model, care must be taken to ensure that the series is in fact convergent and the numerical approximation error created by truncating the series is sufficiently small for the intended application. As Roache (1998) points out, there are many cases where subtle issues arise with the numerical evaluation of infinite series solutions, so they should be used with caution.

6.5.2 Reduction to ordinary differential equations

In some cases, a suitable transformation can be found which reduces a system of PDEs to a system of ordinary differential equations. Methods are available to compute highly-accurate numerical or series solutions for ordinary differential equations. One example is the well-known Blasius solution to the laminar boundary layer equations in fluid dynamics (Schetz, 1993). This solution employs similarity transformations to reduce the incompressible

boundary layer equations for conservation of mass and momentum to a single, nonlinear ordinary differential equation, which can then be accurately solved using series solution (the original approach of Blasius) or by numerical approximation. Consider the situation where a code based on the full Navier–Stokes equation is used to solve for the laminar boundary layer flow over a flat plate. In this case, the solution from the Navier–Stokes code would not converge to the Blasius solution since these are two different mathematical models; the Navier–Stokes equations contain terms omitted from the boundary layer equations which are expected to become important near the leading edge singularity.

6.5.3 Benchmark numerical solutions

Another approximate solution method is to compute a *benchmark numerical solution* with a high-degree of numerical accuracy. In order for a numerical solution to a complex PDE to qualify as a benchmark solution, the problem statement, numerical scheme, and numerical solution accuracy should be documented (Oberkampf and Trucano, 2008). Quantifying the numerical accuracy of benchmark numerical solutions is often difficult, and at a minimum should include evidence that (1) the asymptotic convergence range has been achieved for the benchmark problem and (2) the code used to generate the benchmark solution has passed the order of accuracy code verification test for the options exercised in the benchmark problem. Extensive benchmark numerical solutions for solid mechanics applications are discussed in Oberkampf and Trucano (2008).

6.5.4 Example series solution: 2-D steady heat conduction

The problem of interest in this example is steady-state heat conduction in an infinitely long bar with a rectangular cross section of width L and height H (Dowding, 2008). A schematic of the problem is given in Figure 6.19. If constant thermal conductivity is assumed, then the conservation of energy equation reduces to a Poisson equation for temperature,

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = \frac{-\dot{g}'''}{k}, \tag{6.38}$$

where T is the temperature, k is the thermal conductivity, and $\dot{g}'''$ is an energy source term. The bottom and left boundaries employ zero heat flux (Neumann) boundary conditions, the right boundary a fixed temperature (Dirichlet) boundary condition, and the top boundary is a convective heat transfer (Robin) boundary condition. These boundary conditions are also given in Figure 6.19 and can be summarized as

$$\begin{aligned}
&\frac{\partial T}{\partial x}(0, y) = 0, \\
&\frac{\partial T}{\partial y}(x, 0) = 0, \\
&T(L, y) = T_\infty, \\
&-k\frac{\partial T}{\partial y}(x, H) = h\,[T(x, H) - T_\infty],
\end{aligned} \tag{6.39}$$

where h is the film coefficient from convective cooling.

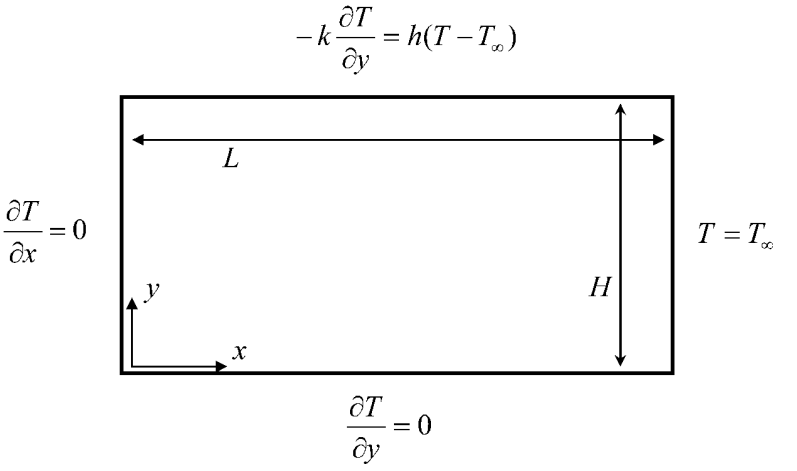

Figure 6.19 Schematic of heat conduction problem in an infinitely long bar of rectangular cross section (Dowding, 2008).

In order to make the Dirichlet boundary condition homogeneous (i.e., equal to zero), the following simple transformation is used:

$$\omega(x, y) = T(x, y) - T_\infty. \tag{6.40}$$

Note that the use of this transformation does not change the form of the governing equation, which can be rewritten in terms of ω as

$$\frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2} = \frac{-\dot{g}'''}{k}. \tag{6.41}$$

The problem statement in terms of $\omega(x, y)$ is shown in Figure 6.20.

The solution to the transformed problem in terms of ω can be found using separation of variables and is

$$\frac{\omega(x, y)}{\frac{\dot{g}''' L^2}{k}} = \frac{1}{2}\left(1 - \frac{x^2}{L^2}\right) + 2\sum_{n=1}^{\infty} \frac{(-1)^n}{\mu_n^3} \frac{\cosh\left(\frac{\mu_n y}{aH}\right)\cos\left(\mu_n \frac{x}{L}\right)}{\frac{1}{Bi}\frac{\mu_n}{a}\sinh\left(\frac{\mu_n}{a}\right) + \cosh\left(\frac{\mu_n}{a}\right)}, \tag{6.42}$$

where the eigenvalues μ_n are given by

$$\mu_n = (2n - 1)\frac{\pi}{2}, \quad n = 1, 2, 3, \ldots \quad \text{and} \cos(\mu_n) = 0,$$

and the constant a and the Biot number Bi are defined as:

$$a = \frac{L}{H}, \quad Bi = \frac{hH}{k}.$$

The infinite series is found to converge rapidly everywhere except near the top wall, where over 100 terms are needed to obtain accuracies of approximately seven significant figures (Dowding, 2008). The following parameters have been used to generate the exact solution given in Figure 6.21, which is presented in terms of the temperature by using the simple

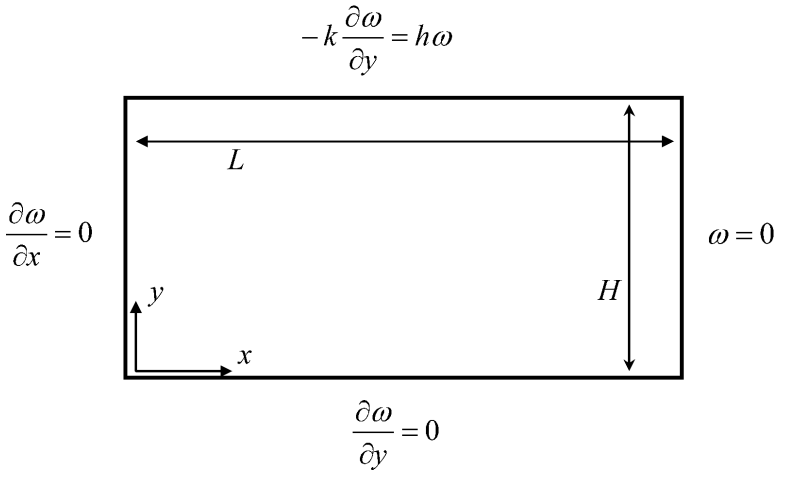

Figure 6.20 Schematic of heat conduction problem in terms of ω (Dowding, 2008).

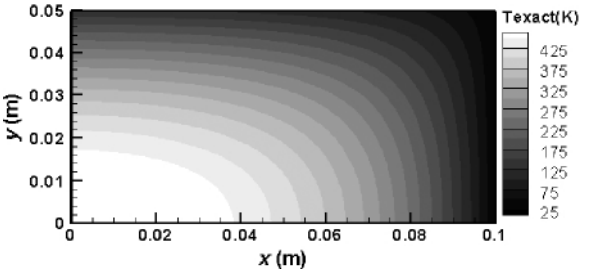

Figure 6.21 Infinite series solution for 2-D steady-state heat conduction.

transformation from Eq. (6.40):

$$\begin{aligned} \dot{g}''' &= 135\,300 \text{ W/m}^3, \\ k &= 0.4 \text{ W/(m} \cdot \text{K)}, \\ L &= 0.1 \text{ m}, \\ H &= 0.05 \text{ m}, \\ T_\infty &= 25 \text{ K}. \end{aligned}$$

These parameters correspond to an aspect ratio of 2, a Biot number of 7.5, and a dimensionless heat source $\dot{g}''' L^2/k$ of 3 382.5.

6.5.5 Example benchmark convergence test: 2-D hypersonic flow

An example of benchmark numerical solutions used with the convergence test for code verification is given by Roy *et al*. (2003). They considered the Mach 8 inviscid flow

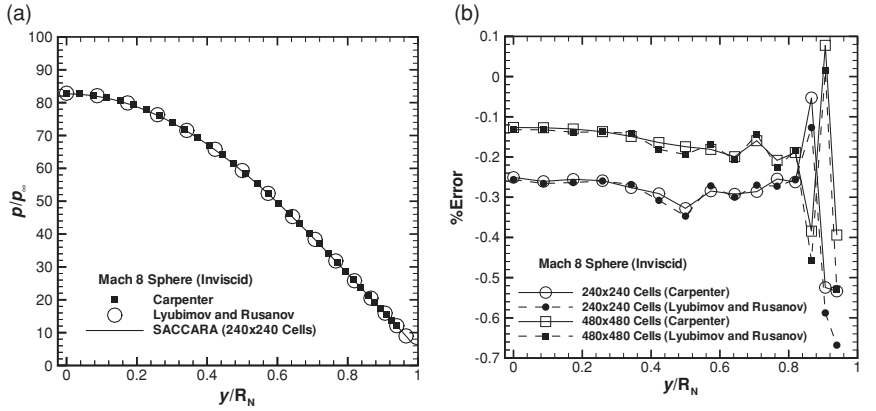

Figure 6.22 Compressible computational fluid dynamics predictions for the Mach 8 flow over a sphere: (a) surface pressure distributions and (b) discretization error in the surface pressure from the SACCARA code relative to the two benchmark solutions (from Roy *et al*., 2003).

of a calorically perfect gas over a sphere-cone geometry. This flow is governed by the Euler equations in axisymmetric coordinates. Two benchmark numerical solutions were employed: a higher-order spectral solution (Carpenter *et al*., 1994) and an accurate finite-difference solution (Lyubimov and Rusanov, 1973). Numerical solutions for surface pressure were computed using the compressible computational fluid dynamics code SACCARA (see Roy *et al*., 2003 for details) and are compared to these two benchmark solutions on the spherical nose region in Figure 6.22. While the pressure distributions appear identical in Figure 6.22a, examination of the discretization error in the SACCARA solution relative to the benchmark solutions in Figure 6.22b shows that the discretization error is small (less than 0.7%) and that it decreases by approximately a factor of two with mesh refinement (i.e., the numerical solutions are convergent). Variations are seen in the discretization error near the sphere-cone tangency point due to the presence of the geometric singularity in the computational fluid dynamics solution (a discontinuity in surface curvature). While these results do demonstrate that the SACCARA code has passed the convergence test, it is generally difficult to assess the order of accuracy using benchmark numerical solutions. A similar benchmark solution for the Navier–Stokes equations involving viscous laminar flow can also be found in Roy *et al*. (2003).

6.6 References

Ames, W. F. (1965). *Nonlinear Partial Differential Equations in Engineering*, New York, Academic Press Inc.

Benton, E. R. and G. W. Platzman (1972). A table of solutions of the one-dimensional Burgers' equation, *Quarterly of Applied Mathematics*. **30**(2), 195–212.

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

Carpenter, M. H., H. L. Atkins, and D. J. Singh (1994). Characteristic and finite-wave shock-fitting boundary conditions for Chebyshev methods, In *Transition, Turbulence, and Combustion*, eds. M. Y. Hussaini, T. B. Gatski, and T. L. Jackson, Vol. 2, Norwell, MA, Kluwer Academic, pp. 301–312.

Carslaw, H. S. and J. C. Jaeger (1959). *Conduction of Heat in Solids*, 2nd edn., Oxford, Clarendon Press.

Chorin, A. J. (1967). A numerical method for solving incompressible viscous flow problems, *Journal of Computational Physics*. **2**(1), 12–26.

Dowding, K. (2008). Private communication, January 8, 2008.

Eca, L. and M. Hoekstra (2006). Verification of turbulence models with a manufactured solution, *European Conference on Computational Fluid Dynamics, ECCOMAS CFD 2006*, Wesseling, P., Onate, E., and Periaux, J. (eds.), Egmond ann Zee, The Netherlands, ECCOMAS.

Eca, L., M. Hoekstra, A. Hay, and D. Pelletier (2007). On the construction of manufactured solutions for one and two-equation eddy-viscosity models, *International Journal for Numerical Methods in Fluids*. **54**(2), 119–154.

Elishakoff, I. (2004). *Eigenvalues of Inhomogeneous Structures: Unusual Closed-Form Solutions*, Boca Raton, FL, CRC Press.

Gottlieb, J. J. and C. P. T. Groth (1988). Assessment of Riemann solvers for unsteady one-dimensional inviscid flows of perfect gases, *Journal of Computational Physics*. **78**(2), 437–458.

Hirsch, C. (2007). *Numerical Computation of Internal and External Flows: Fundamentals of Computational Fluid Dynamics*, 2nd edn., Oxford, Butterworth-Heinemann.

Hopkins, M. M. and C. J. Roy (2004) Introducing the method of nearby problems, *European Congress on Computational Methods in Applied Sciences and Engineering, ECCOMAS 2004*, P. Neittaanmaki, T. Rossi, S. Korotov, E. Onate, J. Periaux, and D. Knorzer (eds.), University of Jyväskylä (Jyvaskyla), Jyväskylä, Finland, July 2004.

Jameson, A., W. Schmidt, and E. Turkel (1981). *Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes*, AIAA Paper 81–1259.

Junkins, J. L., G. W. Miller, and J. R. Jancaitis (1973). A weighting function approach to modeling of irregular surfaces, *Journal of Geophysical Research*. **78**(11), 1794–1803.

Kausel, E. (2006). *Fundamental Solutions in Elastodynamics: a Compendium*, New York, Cambridge University Press.

Kevorkian, J. (2000). *Partial Differential Equations: Analytical Solution Techniques*, 2nd edn., Texts in Applied Mathematics, 35, New York, Springer.

Knupp, P. and K. Salari (2003). *Verification of Computer Codes in Computational Science and Engineering*, K. H. Rosen (ed.), Boca Raton, Chapman and Hall/CRC.

Lyubimov, A. N. and V. V. Rusanov (1973). *Gas Flows Past Blunt Bodies, Part II: Tables of the Gasdynamic Functions*, NASA TT F-715.

Meleshko, S. V. (2005). *Methods of Constructing Exact Solutions of Partial Differential Equations: Mathematical and Analytic Techniques with Applications to Engineering*, New York, Springer.

Oberkampf, W. L. and T. G. Trucano (2008). Verification and validation benchmarks, *Nuclear Engineering and Design*. **238**(3), 716–743.

Oberkampf, W. L., F. G. Blottner, and D. P. Aeschliman (1995). *Methodology for Computational Fluid Dynamics Code Verification/Validation*, AIAA Paper 95–2226

(see also Oberkampf, W. L. and Blottner, F. G. (1998). Issues in computational fluid dynamics code verification and validation, *AIAA Journal*. **36**(5), 687–695).
O'Neil, P. V. (2003). *Advanced Engineering Mathematics*, 5th edn., Pacific Grove, CA, Thomson Brooks/Cole.
Panton, R. L. (2005). *Incompressible Flow*, Hoboken, NJ, Wiley.
Pelletier, D., E. Turgeon, and D. Tremblay (2004). Verification and validation of impinging round jet simulations using an adaptive FEM, *International Journal for Numerical Methods in Fluids*. **44**, 737–763.
Polyanin, A. D. (2002). *Handbook of Linear Partial Differential Equations for Engineers and Scientists*, Boca Raton, FL, Chapman and Hall/CRC.
Polyanin, A. D. and V. F. Zaitsev (2003). *Handbook of Exact Solutions for Ordinary Differential Equations*, 2nd edn., Boca Raton, FL, Chapman and Hall/CRC.
Polyanin, A. D. and V. F. Zaitsev (2004). *Handbook of Nonlinear Partial Differential Equations*, Boca Raton, FL, Chapman and Hall/CRC.
Powers, J. M. and T. D. Aslam (2006). Exact solution for multidimensional compressible reactive flow for verifying numerical algorithms, *AIAA Journal*. **44**(2), 337–344.
Powers, J. M. and D. S. Stewart (1992). Approximate solutions for oblique detonations in the hypersonic limit, *AIAA Journal*. **30**(3), 726–736.
Roache, P. J. (1998). *Verification and Validation in Computational Science and Engineering*, Albuquerque, NM, Hermosa Publishers.
Roache, P. J. (2002). Code verification by the method of manufactured solutions, *Journal of Fluids Engineering*. **124**(1), 4–10.
Roache, P. J. and S. Steinberg (1984). Symbolic manipulation and computational fluid dynamics, *AIAA Journal*. **22**(10), 1390–1394.
Roache, P. J., P. M. Knupp, S. Steinberg, and R. L. Blaine (1990). Experience with benchmark test cases for groundwater flow. In *Benchmark Test Cases for Computational Fluid Dynamics*, I. Celik and C. J. Freitas (eds.), New York, American Society of Mechanical Engineers, Fluids Engineering Division, Vol. **93**, Book No. H00598, pp. 49–56.
Roe, P. L. (1981). Approximate Riemann solvers, parameter vectors, and difference schemes, *Journal of Computational Physics*. **43**, 357–372.
Roy, C. J. (2005). Review of code and solution verification procedures for computational simulation, *Journal of Computational Physics*. **205**(1), 131–156.
Roy, C. J. and M. M. Hopkins (2003). *Discretization Error Estimates using Exact Solutions to Nearby Problems*, AIAA Paper 2003–0629.
Roy, C. J. and A. J. Sinclair (2009). On the generation of exact solutions for evaluating numerical schemes and estimating discretization error, *Journal of Computational Physics*. **228**(5), 1790–1802.
Roy, C. J., M. A. McWherter-Payne, and W. L. Oberkampf (2003). Verification and validation for laminar hypersonic flowfields Part 1: verification, *AIAA Journal*. **41**(10), 1934–1943.
Roy, C. J., C. C. Nelson, T. M. Smith, and C.C. Ober (2004). Verification of Euler/Navier–Stokes codes using the method of manufactured solutions, *International Journal for Numerical Methods in Fluids*. **44**(6), 599–620.
Roy, C. J., A. Raju, and M. M. Hopkins (2007a). Estimation of discretization errors using the method of nearby problems, *AIAA Journal*. **45**(6), 1232–1243.
Roy, C. J., E. Tendean, S. P. Veluri, R. Rifki, E. A. Luke, and S. Hebert (2007b). *Verification of RANS Turbulence Models in Loci-CHEM using the Method of Manufactured Solutions*, AIAA Paper 2007–4203.

Schetz, J. A. (1993). *Boundary Layer Analysis*, Upper Saddle River, NJ, Prentice-Hall.
Seidel, G. D. (2009). Private communication, November 6, 2009.
Shih, T. M. (1985). Procedure to debug computer programs, *International Journal for Numerical Methods in Engineering*. **21**(6), 1027–1037.
Slaughter, W. S. (2002). *The Linearized Theory of Elasticity*, Boston, MA, Birkhauser.
Steinberg, S. and P. J. Roache (1985). Symbolic manipulation and computational fluid dynamics, *Journal of Computational Physics*. **57**(2), 251–284.
Stetter, H. J. (1978). The defect correction principle and discretization methods, *Numerische Mathematik*. **29**(4), 425–443.
Tannehill, J. C., D. A. Anderson, and R. H. Pletcher (1997). *Computational Fluid Mechanics and Heat Transfer*, 2nd edn., Philadelphia, PA, Taylor and Francis.
Thompson, J. F., Z. U. A. Warsi, and C. W. Mastin (1985). *Numerical Grid Generation: Foundations and Applications*, New York, Elsevier. (www.erc.msstate.edu/publications/gridbook)
Timoshenko, S. P. and J. N. Goodier (1969). *Theory of Elasticity*, 3rd edn., New York, McGraw-Hill.
Tremblay, D., S. Etienne, and D. Pelletier (2006). *Code Verification and the Method of Manufactured Solutions for Fluid–Structure Interaction Problems*, AIAA Paper 2006–3218.
White, F. M. (2006) *Viscous Fluid Flow*, New York, McGraw-Hill.
Yanenko, N. N. (1964). Compatibility theory and methods of integrating systems of nonlinear partial differential equations, *Proceedings of the Fourth All-Union Mathematics Congress*, Vol. **2**, Leningrad, Nauka, pp. 613–621.
Zadunaisky, P. E. (1976). On the estimation of errors propagated in the numerical integration of ordinary differential equations, *Numerische Mathematik*. **27**(1), 21–39.