

# Verification

Christopher J. Roy

*Aerospace and Ocean Engineering Department*

*Virginia Tech*

*215 Randolph Hall*

*Blacksburg, Virginia 24061 USA*

*[cjroy@vt.edu](mailto:cjroy@vt.edu)*

**Synonyms:** Code Verification, Solution Verification, Calculation Verification, Numerical Uncertainty

**Short Definition:** Verification is the process of ensuring that numerical simulation results are a sufficiently accurate representation of the exact solution to a mathematical model.

## 1. Introduction

Mathematical *models* are used in science and engineering to describe the behavior of a system. In many cases, these models take the form of partial differential equations (PDEs) which require numerical solutions (i.e., *simulations*) due to their complexity. Verification and validation provide a means for assessing the credibility and accuracy of mathematical models and their subsequent simulations [1-3]. *Verification* deals with assessing the numerical accuracy of a simulation relative to the true result of the model. *Validation*, on the other hand, is the assessment of the accuracy of the model relative to experimental observations. For models based on PDEs (which we will consider exclusively in this article), the verification process involves a number of steps as shown in Figure 1. Starting with a PDE-based mathematical model, one must first choose the discretization algorithm (e.g., finite difference, finite volume, finite element). Next, this algorithm must be implemented in the computational mathematics software. Finally, the software is used, along with appropriate grids, time steps, iterative tolerances, etc., to produce a simulation prediction. The first two steps in this process require that evidence be gathered that there are no algorithm inconsistencies or coding mistakes and is called *code verification*. The last step involves the quantitative estimation of the numerical errors that arise during the simulation process and is termed *solution verification*. The end result from the verification process is an estimated numerical uncertainty in the simulation predictions that can be used in assessing the overall prediction uncertainty.

## 2. Code Verification

Code verification ensures that the computational software is an accurate representation of the underlying PDEs. It is accomplished by employing appropriate software engineering practices and by using code order of accuracy verification to ensure that there are no mistakes in the computer code or inconsistencies in the discrete algorithm. While software engineering is a vast subject unto itself, some

aspects that are particularly useful for computational mathematics software include version control, static analysis, dynamic testing, and regression testing (see Ref. [2] for more details). Before proceeding with a discussion of code verification, it is important to identify what simulation output quantities should be evaluated. In general, one should examine error norms in the solution dependent variables (typically  $L_1$ ,  $L_2$ , and  $L_\infty$  norms) as well as any global System Response Quantities (SRQs) that one is interested in predicting. For example, the discrete  $L_2$  norm appropriate for a finite volume solution can be computed as

$$\|u - \tilde{u}\|_2 = \left[ \frac{1}{\Omega} \sum_{n=1}^N \omega_n |u_n - \tilde{u}_n|^2 \right]^{1/2} \quad (1)$$

where  $u_n$  is the discrete solution in cell  $n$ ,  $\tilde{u}$  is the exact solution to the PDEs, and  $\Omega$  is the total volume of the domain.

## 2.1 Order of accuracy testing

There are various criteria that can be used during code verification. However, the most rigorous code verification criterion is the order of accuracy test, where one assesses whether the numerical solutions converge to the exact solution to the PDEs at the expected rate (i.e., the *formal order of accuracy*) for the discrete algorithm. The formal order of accuracy of an algorithm is commonly found from a truncation error analysis which addresses the convergence of the discrete equations to the PDEs; however, there are pitfalls with using this approach, especially on unstructured grids [2]. For consistent algorithms, the truncation error will be proportional to the discretization parameters (e.g., spatial element size  $\Delta x$ , time step size  $\Delta t$ ) to some exponents which usually correspond to the formal order of accuracy of the discretization scheme. For example, consider a simple Euler explicit finite difference discretization of the 1D heat equation with diffusivity  $\alpha$  on a uniform mesh with node spacing  $\Delta x$ . The leading truncation error terms at any node  $i$  and time step  $n$  are

$$TE_i^n = \left[ \frac{1}{2} \frac{\partial^2 T}{\partial t^2} \right]_i^n \Delta t^1 - \left[ \frac{\alpha}{12} \frac{\partial^4 T}{\partial x^4} \right]_i^n \Delta x^2 + O(\Delta t^2, \Delta x^4) \quad (2)$$

which indicates that the discretization scheme is formally first order accurate in time and second order accurate in space.

The *observed order of accuracy* is the actual rate at which the numerical solutions converge to the exact solution to the PDEs with systematic refinement of the mesh and/or time step. Consider a power series expansion of some SRQ in terms of a generic discretization parameter  $h$  about the exact solution to the PDEs  $\tilde{f}$ :

$$f_h = \tilde{f} + g_p h^p + O(h^{p+1}). \quad (3)$$

A similar expansion with a discretization parameter that is  $r$  times larger (e.g.,  $r = h_{\text{coarse}} / h_{\text{fine}}$  is the grid refinement factor) gives:

$$f_{rh} = \tilde{f} + g_p (rh)^p + O(h^{p+1}). \quad (4)$$

Combining these two expressions, neglecting the higher-order terms, and solving for  $p$  yields an expression for the observed order of accuracy  $\hat{p}$ :

$$\hat{p} = \frac{\ln\left(\frac{f_{rh} - \tilde{f}}{f_h - \tilde{f}}\right)}{\ln(r)} \quad (5)$$

where  $f_{rh}$  and  $f_h$  are the coarse and fine grid SRQs, respectively. This expression for the observed order of accuracy requires solutions on two mesh levels as well as knowledge of the exact solution to the PDEs  $\tilde{f}$ .

The order of accuracy test examines the limiting behavior of  $\hat{p}$  to ensure that it approaches the formal order of accuracy with systematic mesh refinement (see below). In addition to mistakes in the software programming, the following conditions are required to pass the order of accuracy test. First, the iterative and round-off errors in the numerical solution must be significantly less than the fine grid discretization error (typically two orders of magnitude). Second, the mesh and time step must be sufficiently small so that the lowest order terms in the truncation and discretization error expansions dominate the higher-order terms (i.e., the numerical solutions must be in the asymptotic convergence range). Third, the mesh and time step must be systematically refined as discussed in the next section.

## 2.2 Systematic mesh refinement

*Systematic mesh refinement* [2] requires that the mesh be refined uniformly by a factor  $h$  in each coordinate direction, e.g.,

$$h = \frac{\Delta x}{\Delta x_{\text{ref}}} = \frac{\Delta y}{\Delta y_{\text{ref}}} = \frac{\Delta z}{\Delta z_{\text{ref}}} \quad (6)$$

and that the mesh quality be constant or improve with mesh refinement. Ensuring systematic mesh refinement can be challenging, especially for unstructured meshes which contain more than one type of mesh topology (e.g., hexahedral, tetrahedral, and prismatic elements). For structured grids, where grid transformations can be used to transform the grid to a Cartesian computational space, systematic refinement can be ensured by starting with the fine grid and removing every other grid line, resulting in a grid refinement factor of  $r = 2$ . The drawback to this approach is that each level of refinement requires a factor of 8 increase in cells/elements for three dimensional problems.

## 2.3 Exact solutions

Rigorous code order of accuracy testing requires an exact solution to the PDEs. Traditional methods of obtaining exact solutions to PDEs (e.g., separation of variables, series solutions, transformations) often fail for PDEs involving complicated geometry, nonlinearity, non-constant coefficients, complicated sub-models, and/or multi-physics coupling. When exact solutions are found for complex equations, they often depend on significant simplifications in dimensionality, geometry, physics, etc.

An alternative to the traditional approach for obtaining exact solutions to PDEs is the Method of Manufactured Solutions (MMS). The concept behind MMS is to take an original PDE and modify it by appending an analytic source term so that it satisfies a chosen (usually non-physical) solution. Consider an original PDE with dependent variable  $u$  written in operator notation as  $L(u) = 0$ . Next, choose an analytic manufactured solution  $\hat{u}$  which has non-trivial analytic derivatives. The PDE is then operated onto the manufactured solution in order to obtain the analytic source term:  $s = L(\hat{u})$ . The modified PDE is found by appending this source term to the original PDE  $L(u) = s$ , which will be exactly solved by the chosen manufactured solution  $\hat{u}$ .

Manufactured solutions should be chosen to be analytic functions with smooth derivatives, and it is important to ensure that all of the derivatives appearing in the PDE are non-zero. Trigonometric and exponential functions are recommended since they are smooth and infinitely differentiable. Although the manufactured solutions do not need to be physically realistic when used for code verification, they should be chosen to obey the physical constraints that are embodied in the code (e.g., positive temperatures, species concentrations). Finally, care should be taken that one term in the PDEs does not dominate the other terms. For example, even if the applications of interest for a Navier-Stokes code are high-Reynolds number flows, when performing code verification studies, the manufactured solution should be chosen to give Reynolds numbers of order unity so that convective and diffusive terms will be the same order of magnitude.

As an example of order verification using MMS [2], consider steady-state heat conduction with a constant thermal diffusivity, which reduces to Poisson's equation for the temperature  $T$ :

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = s(x, y) \quad (7)$$

where  $s(x, y)$  is the manufactured solution source term. The following manufactured solution is chosen

$$\hat{T}(x, y) = T_0 + T_x \cos\left(\frac{a_x \pi x}{L}\right) + T_y \sin\left(\frac{a_y \pi y}{L}\right) + T_{xy} \sin\left(\frac{a_{xy} \pi xy}{L^2}\right) \quad (8)$$

where

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

and Dirichlet (fixed-value) boundary conditions are applied on all four boundaries as determined by Equation (8). A family of stretched Cartesian meshes is created by first generating the finest mesh (129×129 nodes), and then successively eliminating every other gridline to create the coarser meshes, thus ensuring systematic refinement. The manufactured solution from Equation (8) is shown graphically in Figure 2. Discrete  $L_2$  norms of the discretization error (i.e., the difference between the numerical solution and the manufactured solution) are computed for grid levels from 129×129 nodes ( $h = 1$ ) to 9×9 nodes ( $h = 16$ ). The observed order of accuracy of these  $L_2$  norms is computed for successive mesh levels, and the results are shown in Figure 3. 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 exercised.

### 3. Solution Verification

The main focus of solution verification is the estimation of the numerical errors that occur when PDEs are discretized and solved numerically. Numerical errors can arise in computational mathematics due to computer round-off, iteration, and discretization. Round-off errors occur due to the fact that only a finite number of significant figures can be used to store floating point numbers in a digital computer. Round-off errors are usually small, but can be reduced if necessary by increasing the number of significant figures used in floating point computations (e.g., by changing from single to double precision arithmetic). Iterative convergence errors are present when discretization of the PDEs results in a simultaneous set of algebraic equations that are solved approximately or when relaxation techniques are used to obtain a solution. Discretization error arises due to the fact that the spatial domain is decomposed into a finite number of nodes/elements and, for unsteady problems, time is advanced with a finite time step. Iterative and discretization errors are discussed in detail below.

#### 3.1 Iterative error

In computational mathematics, the *iterative error* is the difference between the current approximate solution to the discretized equations and the exact solution to the discretized equations. For a global SRQ  $f$ , we can thus define the iterative error at iteration  $k$  as

$$\mathcal{E}_h^k = f_h^k - f_h \quad (9)$$

where  $h$  refers to the discrete solution on a mesh with discretization parameters ( $\Delta x$ ,  $\Delta y$ ,  $\Delta t$ , etc.) represented collectively by  $h$ ,  $f_h^k$  is the current iterative solution, and  $f_h$  is the exact solution to the discrete equations (not to be confused with the exact solution to the PDEs  $\tilde{f}$ ). We might instead be concerned with the iterative error in the entire solution over the domain (i.e., the dependent variables in

the PDEs), in which case the iterative error for each dependent variable  $u$  should be measured as a norm over the domain.

For stationary iterative methods (e.g., Jacobi, Gauss-Seidel, multigrid) applied to linear systems, iterative convergence is governed by the eigenvalues of the iteration matrix. For linear problems, when the maximum eigenvalue of the iteration matrix is real, the limiting iterative convergence behavior will be monotone. When it is complex, however, the limiting iterative convergence behavior will generally be oscillatory. In these cases, convergence of the iterative method requires that the spectral radius of the iteration matrix be less than one [4]. For nonlinear problems, the linearized system is often not solved to convergence, but only solved for a few iterations (sometimes as few as one) before the nonlinear terms are updated, and the form of the convergence is often much more difficult to characterize.

The discrete equations can be written in the form

$$L_h(u_h) = 0 \quad (10)$$

where  $L_h$  is the linear or nonlinear discrete operator and  $u_h$  is the exact solution to the discrete equations. The *iterative residual* is found by plugging the current iterative solution  $u_h^{k+1}$  into Equation (10), i.e.,

$$\mathfrak{R}_h^{k+1} = L_h(u_h^{k+1}) \quad (11)$$

where  $\mathfrak{R}_h^{k+1} \rightarrow 0$  as  $u_h^{k+1} \rightarrow u_h$ . Although monitoring the iterative residuals often serves as an adequate indication as to whether iterative convergence of the solution has been achieved, it does not by itself provide any guidance as to the size of the *iterative error* in the solution quantities of interest. Since the iterative residual norms have been shown to follow closely with actual iterative errors for many problems [2], a small number of computations should be sufficient to determine how the iterative errors in the SRQ scale with the iterative residuals for the cases of interest.

An example of this procedure is given in Figure 4 for laminar viscous flow through a packed bed of spherical particles [5]. The quantity of interest is the average pressure gradient across the bed, and the desired iterative error level in the pressure gradient is 0.01%. The iterative residuals in the conservation of mass and conservation of momentum equations are first converged to  $10^{-7}$  (relative to their initial levels) then the value of the pressure gradient at this point is taken as an approximation to the exact solution to the discrete equations  $\hat{f}_h$ . The iterative error for all previous iterations is then approximated as  $\varepsilon_h^k \approx f_h^k - \hat{f}_h$ . Figure 4 shows that for the desired iterative error level in the pressure gradient of 0.01%, the iterative residual norms should be converged down to approximately  $10^{-6}$ . Simulations for similar problems can be expected to require approximately the same level of iterative residual convergence in order to achieve the same iterative error tolerance in the pressure gradient.

### 3.2 Discretization error

The *discretization error* is the difference between the exact solution to the discretized equations and the exact solution to the PDEs. It is difficult to estimate for practical problems and is often the largest of the numerical error sources. As shown in Figure 5, methods for estimating discretization error can be broadly categorized as either recovery methods or residual-based estimators [2,6]. Recovery methods involve post-processing of the solution(s) and include Richardson extrapolation [1-3,6], order extrapolation [2,6], and recovery methods from finite elements [7]. The residual-based methods employ additional information about the problem being solved and include discretization error transport equations [2,6], defect correction methods [8], implicit/explicit residual methods in finite elements [2,7], and adjoint methods for estimating the error in solution functionals (i.e., SRQs) [7]. Due to space limitations, we will limit our discussion to Richardson extrapolation.

*Richardson extrapolation* uses solutions on two or more systematically refined meshes to estimate the exact solution to the PDEs, which can in turn be used to provide an error estimate for the numerical solutions. Consider the two series expansions for the numerical solution about the exact solution to the PDEs given earlier by Equations (3) and (4) for systematically-refined meshes with spacing  $h$  and  $rh$ , respectively. Assuming for now that the solutions are in the asymptotic range (i.e., that the observed order of accuracy matches the formal order), these two equations can be solved for an estimate of the exact solution to the PDEs by neglecting the higher-order terms to obtain the Richardson extrapolation estimate

$$\bar{f} = f_h + \frac{f_h - f_{rh}}{r^p - 1} \quad (12)$$

which is generally a  $(p+1)$ -order accurate estimate of the exact solution to the PDEs  $\tilde{f}$ . It can be used to estimate the discretization error in the fine grid solution, i.e.,  $\bar{\epsilon}_h = f_h - \tilde{f}$ , resulting in the error estimate:

$$\bar{\epsilon}_h = \frac{f_{rh} - f_h}{r^p - 1}. \quad (13)$$

Note that in addition to the assumption that both solutions are in the asymptotic range, this error estimate will be accurate only when iterative and round-off errors are much smaller than the fine grid discretization error.

Regardless of the approach used for estimating the discretization error, the *reliability* of the discretization error estimate depends on the solutions being in the asymptotic mesh convergence range, which is extremely difficult to achieve for complex computational mathematics applications. Verifying that the solutions are in the asymptotic range can be done by computing the observed order of accuracy using numerical solutions on three systematically-refined meshes. For systematic refinement by the factor  $r$ , one has  $h_{fine} = h$ ,  $h_{medium} = rh$ , and  $h_{coarse} = r^2h$  and the observed order of accuracy can be found as [1]:

$$\hat{p} = \frac{\ln\left(\frac{f_{r^2h} - f_{rh}}{f_{rh} - f_h}\right)}{\ln(r)}. \quad (14)$$

The case when the grid refinement factor between the fine and medium meshes differs from that between the medium and coarse meshes is addressed in [1]. Note that the observed order of accuracy will only match the formal order when all three grid levels are in the asymptotic range. A similar expression for the observed order of accuracy can be derived in terms of the error estimates found on two systematically-refined meshes (e.g., for use with residual-based error estimators):

$$\hat{p} = \frac{\ln\left(\frac{\bar{\varepsilon}_{rh}}{\bar{\varepsilon}_h}\right)}{\ln(r)}. \quad (15)$$

#### 4. Numerical Uncertainty

In some cases, when numerical errors can be estimated with a high degree of confidence, they can be removed from the numerical solution – a process similar to that used for well-characterized bias errors in an experiment. More often, however, the numerical errors are estimated with significantly less certainty, and thus they should be converted into numerical uncertainties, with the uncertainty coming from the error estimation process itself [2,9]. One of the simplest methods for converting an error estimate to an uncertainty is to use the magnitude of the error estimate to apply uncertainty bands about the simulation prediction, possibly with an additional factor of safety included. For example, the Richardson extrapolation estimate of discretization error  $\bar{\varepsilon}_h$  discussed above can be converted to a numerical uncertainty  $U_{\text{Discretization}}$  as

$$U_{\text{Discretization}} = F_s |\bar{\varepsilon}_h| \quad (16)$$

where  $F_s \geq 1$  is the factor of safety. The resulting interval for the numerical solution, accounting for numerical uncertainties, can be approximated by applying this uncertainty to the fine grid solution

$$f_h \pm U_{\text{Discretization}} = f_h \pm F_s |\bar{\varepsilon}_h|. \quad (17)$$

These concepts are shown graphically in Figure 6 with a factor of safety of approximately  $F_s = 1.5$ . When the error estimate is poor (i.e., when the true model solution  $\tilde{f}$  differs significantly from the estimated model solution  $\bar{f}$ , as suggested by the figure), this approach is designed to still potentially provide conservative numerical uncertainty estimates, depending of course on the chosen factor of safety. It is recommended that this uncertainty be centered about the numerical solution  $f_h$  rather than the estimated



exact solution  $\bar{f}$  since the latter can lead to erroneous (and possibly physically non-realizable) values. When multiple sources of numerical error are present, then a conservative approach is to simply add the numerical uncertainties together [2,9], i.e.,

$$U_{\text{NUM}} = U_{\text{RoundOff}} + U_{\text{Iteration}} + U_{\text{Discretization}} \quad (18)$$

While numerical uncertainties are epistemic (i.e., due to a lack of knowledge rather than inherent randomness), it is currently an open question as to whether these uncertainties should be characterized probabilistically or in some other fashion (e.g., as intervals) [9,10].

## References

1. Roache, P. J. (2009), *Fundamentals of Verification and Validation*, Hermosa Publishers, Socorro, New Mexico.
2. Oberkampf, W. L. and C. J. Roy (2010), *Verification and Validation in Scientific Computing*, Cambridge University Press, Cambridge.
3. Roy, C. J. (2005), "Review of code and solution verification procedures for computational simulation," *Journal of Computational Physics*, Vol. 205, pp. 131-156.
4. Golub, G. H. and C. F. Van Loan (1996). *Matrix Computations*, 3rd edn., Baltimore, The Johns Hopkins University Press
5. Duggirala, R., C. J. Roy, S. M. Saeidi, J. Khodadadi, D. Cahela, and B. Tatarchuck (2008). Pressure drop predictions for microfibrinous flows using CFD, *Journal of Fluids Engineering*. 130(DOI: 10.1115/1.2948363).
6. Roy, C. J. (2010), "Review of discretization error estimators in scientific computing," AIAA Paper 2010-0126.
7. Ainsworth, M. and Oden, J. T. (2000). *A Posteriori Error Estimation in Finite Element Analysis*, Wiley Interscience, New York.
8. Skeel, R.D. (1986). "Thirteen Ways to Estimate Global Error," *Numerische Mathematik*, Vol. 48, pp. 1-20.
9. Roy, C. J. and W. L. Oberkampf (2011). "A Comprehensive Framework for Verification, Validation, and Uncertainty Quantification in Scientific Computing," *Computer Methods in Applied Mechanics and Engineering*, Vol. 200, Nos. 25-28, pp. 2131-2144 (DOI: 10.1016/j.cma.2011.03.016).
10. Roy, C. J. and Balch, M. S. (2012), "A Holistic Approach to Uncertainty Quantification with Application to Supersonic Nozzle Thrust," accepted for publication in the *International Journal for Uncertainty Quantification*, January 2012.

## Figures

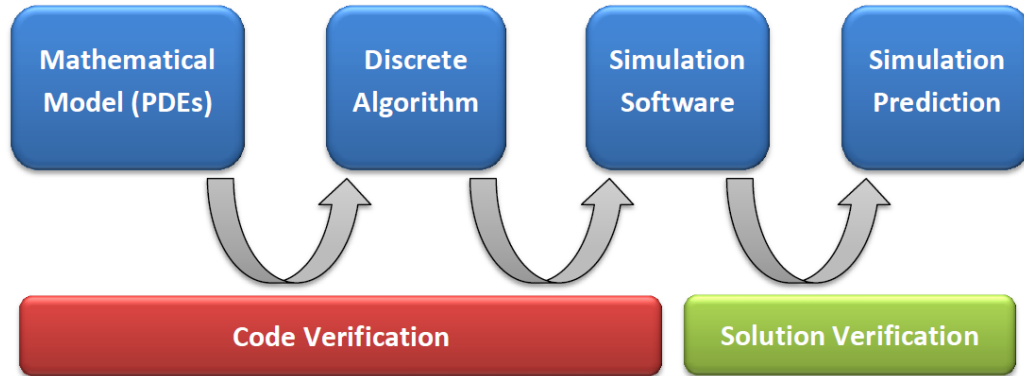


Figure 1. Schematic of the simulation process showing code and solution verification activities.

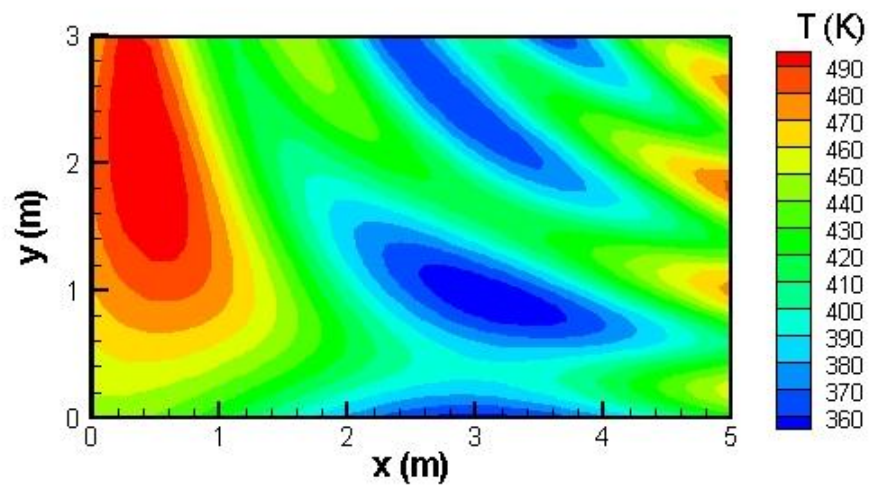


Figure 2. Manufactured solution for temperature for the 2D steady heat conduction problem

(reproduced from Ref. [2]).

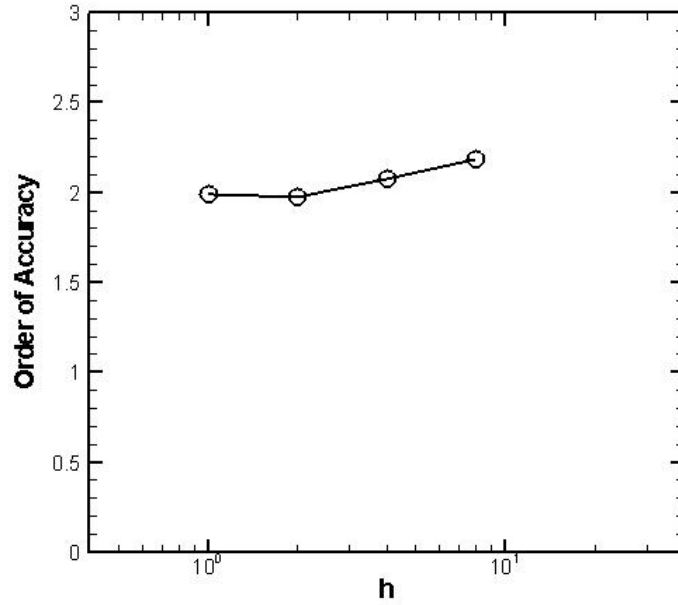


Figure 3. Observed order of accuracy for the 2D steady heat conduction problem (reproduced from Ref. [2]).

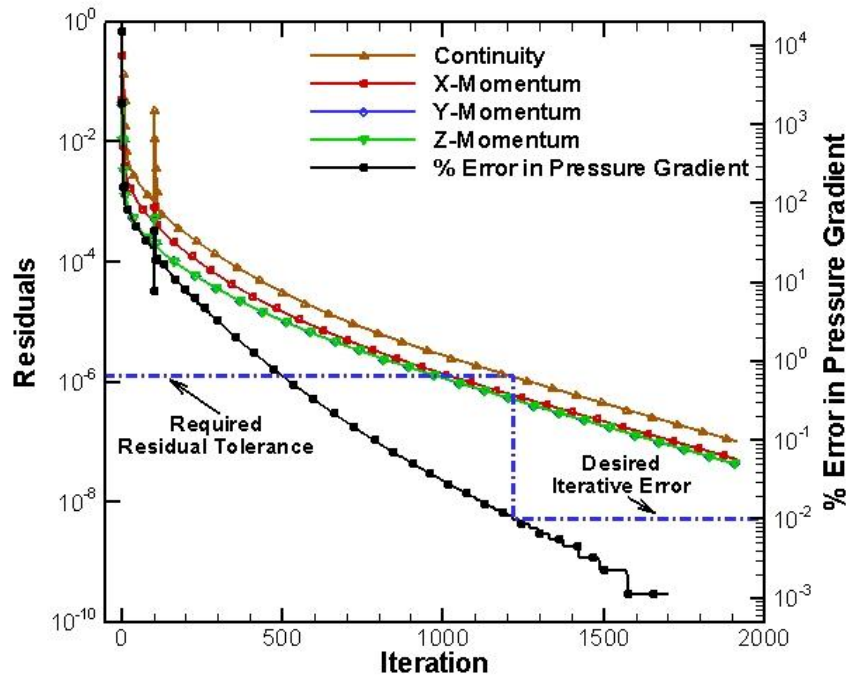


Figure 4. Norms of the iterative residuals (left axis) and percentage error in pressure gradient (right axis) for laminar flow through a packed bed (reproduced from Ref. [5]).

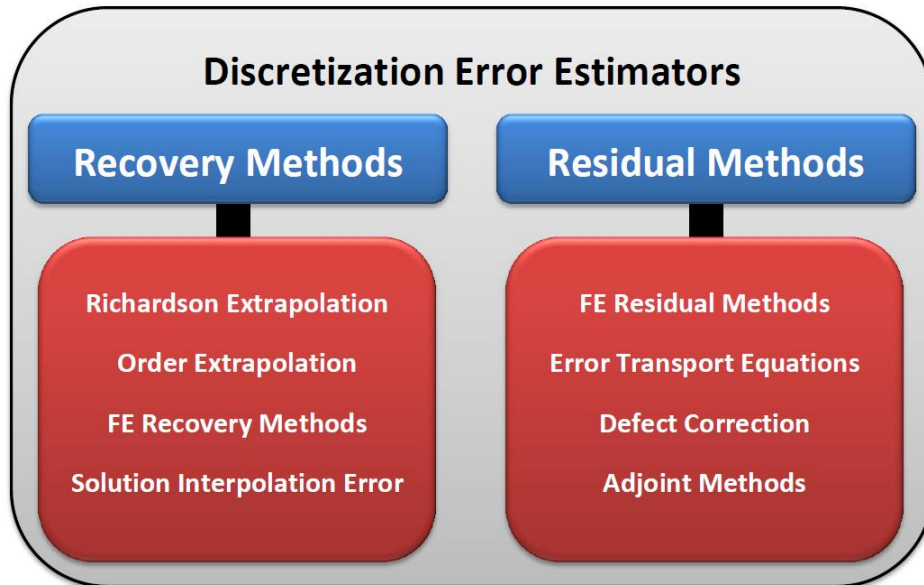


Figure 5. Overview of discretization error estimation approaches.

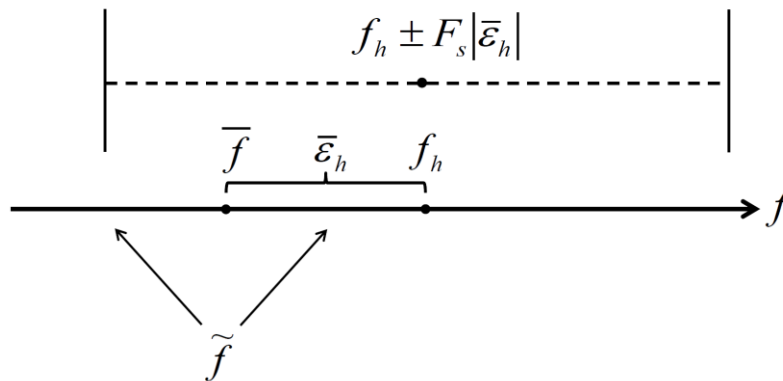


Figure 6. Example of converting a discretization error estimate to a numerical uncertainty

(reproduced from Ref. [10]).