CFD Project Combining Solution Algorithms, Software Development/ Validation, and Viscous Flow Calculations

W. R. BRILEY,[1] B. K. HODGE[2]

[1] *Department of Mechanical Engineering and NSF Engineering Research Center for Computational Field Simulation, Mississippi State University, Mississippi State, Mississippi* 39762-5925

[2] *Department of Mechanical Engineering, Mississippi State University, Mississippi State, Mississippi* 39762-5925

Received 27 November 1995; accepted 17 May 1997

ABSTRACT: This project integrates lecture material on computational methodology for viscous flows with hands-on exposure to code development, validation, and case running, all in a context suitable for varying programming skills. Students are given a well-documented, partially completed code for solving the boundary-layer equations. They recreate the missing parts in accordance with material covered in lectures and then compute laminar and turbulent flows. Examples of results are given. © 1997 John Wiley & Sons, Inc. Comput Appl Eng Educ **5:** 161–168, 1997

Keywords: fluid dynamics; viscous flow; computational methodology; solution algorithms; software development/validation; boundary-layer equations

INTRODUCTION

The computational fluid dynamics (CFD) project described here involves numerical solution of the laminar and turbulent boundary-layer equations, as well as an unsteady viscous Burger's equation, which serves as a model for the Navier–Stokes equations. The project is intended for use in the second of a two-course sequence of 3-h graduate-level mechanical engineering courses on viscous flow. The first course (Viscous Flow I) is a relatively standard introductory course in viscous flow geared for first-year graduate students, and covers (1) viscous flow fundamentals, (2) derivations of the Navier–Stokes and boundary-layer equations, (3) classical and numerical solutions of the Navier–Stokes and boundary-layer equations, and (4) introduction to inviscid flow theory. The second course (Viscous Flow II) contains an introduction to turbu-

Correspondence to W. R. Briley.

 CCC 1061-3773/97/030161-08

lence, turbulence modeling, and computational methods for viscous flows and covers (1) flow instability and transition, (2) physical and mathematical description of turbulent mean flows, (3) turbulence modeling, and (4) introduction to numerical methods for the time-averaged boundary-layer and Navier–Stokes equations. This two-course sequence emphasizes viscous flow theory, but is also intended to provide an introduction to modern computational methodology (approximately one-half of the second course). For more depth or specialization in CFD, students can choose from several other graduate courses: CFD I, II, and III; Numerical Heat Transfer; Numerical Grid Generation I and II; and a cross-disciplinary course, Topics in Engineering and High-Performance Computing.

The project is designed to provide meaningful hands-on exposure to numerical methods, their accuracy and efficiency, programming, using and/or modifying existing computer programs, as well as computation and interpretation of laminar and turbulent flow cases. It can be started after only 2 weeks of lectures covering the basics of finite-difference/volume schemes as applied to the laminar boundary-layer equations, and continues concurrently with supporting lecture material throughout the semester. Thus, the students are engaged very quickly, avoiding the problem of waiting to begin a project until much lecture material is covered and thereby leaving insufficient time to complete the project. Another consideration is that projects involving both writing a new viscous flow program and significant case running can require considerable student time, especially since first-year graduate students have varying levels of programming skills. On the other hand, if the project involves only case running using an existing code, then the student is not exposed to the inner workings of the code and has less appreciation of the solution methodology and interpretation of results.

In developing this project and course, the authors took the viewpoint that Navier–Stokes simulations will be increasingly important in flow analysis and design applications in industry, whereas boundary-layer computations have been more or less routine for some time. Accordingly, the numerical content of the course is focused on solving the Navier–Stokes equations. However, the study of boundary layers is thought to be critical for understanding viscous flow behavior in general, and is helpful when computing and/or interpreting Navier–Stokes solutions. Many numerical topics relevant to the Navier–Stokes equations can be handled in the context of the boundary-layer equations, including systems of equations, marching solution of boundary-value problems, linearization, coupling/decoupling, and implicit/explicit algorithms. Although there are no doubt many other ways to structure course offerings which include CFD projects, the current project seems to work well in this context.

Although numerical solution of the boundary-layer and Burger's equations is not a novel exercise, the present project is thought to be innovative in the manner in which an existing code is used to incorporate and integrate the perspectives of users and developers regarding numerical fundamentals, software development/modification, case running/interpretation, and verification/validation, all in a context suitable for varying programming skills.

MOTIVATION FOR THE PROJECT

It is well known that inadequate understanding of vortex shedding behind bluff bodies contributed to the collapse of the Tacoma Narrows Bridge in 1940. This rather spectacular example of an engineering design failure provides motivation for the study of viscous flow. A more recent failure (the sinking and loss of the Sleipner A oil platform in the North Sea in 1991) has been traced to shortcomings in using finite element software to design the platform [1]. This and other such examples should motivate a careful study of computational methodology. Over the years, a number of CFD specialists have encouraged a healthy skepticism of flow simulation codes and their results. Users are sometimes tempted to assume that results are correct if they are not obviously wrong. Error can be difficult to assess in complex flow cases, and it helps for the user to know and understand something about how the software works. Knowledge of the computational methodology, tests for simple cases with known solutions, and validation by comparison with experiment (where feasible) are all helpful before running complex or important cases. For an interesting discussion on this topic, see Petroski [1].

PROJECT SUMMARY

The project begins by giving students a partially completed FORTRAN 77 code for solving the laminar boundary-layer equations. This code (about 200 lines plus extensive commenting) is carefully written in modular form and documented for ease of

understanding. A small number of key lines representing the solution algorithm and some postprocessing have been removed. The students then study the code and complete or recreate the missing parts in accordance with material covered in lectures. Although students have the option of writing their own original code, the exercise with a partially completed code saves a lot of time while retaining the programming/debugging element. Also, engineers often have to modify existing codes they did not write, and this exercise affords some practice. Students are given specific cases to run, initially for a flat plate whose solution is known, and then for other cases. To gain an understanding of internal flows, they then modify the solution algorithm and code to simulate developing flow in a straight channel. Following lectures covering turbulence and turbulence modeling, the students again modify their code to use a nonuniform grid and implement two or three turbulence models, then run various cases relating to accuracy and the effect of mesh spacing. The case running can also have a design-oriented objective such as finding the maximum area ratio for a straight-walled diffuser of fixed length which avoids separation. Later, the code is modified (or students can write their own code) to solve an unsteady viscous Burger's equation using at least two solution algorithms, and cases are run to illustrate issues of numerical accuracy, stability, and iterative convergence rate.

COMPUTATIONAL METHODS USED

The course material is drawn in part from the CFD textbook by Anderson et al. [2], and the material on turbulence and turbulence modeling is taken from texts by Shetz [3] and White [4]. Some of the test cases used here are similar to those of Shetz [3], so the present project would work very well in a course using this text. The computer program has a modular form which can be adapted easily to use methods favored by other instructors.

The governing equations are the continuity and momentum equations for an incompressible steady boundary layer (cf. Shetz [3], p. 103; Anderson et al. [2], p. 331). These equations are solved in nondimensional form using the following fully implicit, sequentially decoupled algorithm (along the lines of Shetz [3], p. 110; or Anderson et al. [2], p. 335):

Momentum:

$$u_j^n\left(\frac{u^{n+1}-u^n}{\Delta x}\right)_j + v_j^n\left(\frac{u_{j+1}-u_{j-1}}{2\Delta y_{j,av}}\right)^{n+1} = U_e^n\left(\frac{U_e^{n+1}-U_e^n}{\Delta x}\right) + \frac{1}{Re\Delta y_{j,av}}\left[\nu_{j,av}^n\left(\frac{u_{j+1}-u_j}{\Delta y_j}\right)^{n+1} - \nu_{j-1,av}\left(\frac{u_j-u_{j-1}}{\Delta y_{j-1}}\right)^{n+1}\right]$$

Continuity:

$$\left(\frac{v_j-v_{j-1}}{\Delta y_{j-1}}\right)^{n+1} + \frac{1}{2}\left[\left(\frac{u^{n+1}-u^n}{\Delta x}\right)_j + \left(\frac{u^{n+1}-u^n}{\Delta x}\right)_{j-1}\right] = 0$$

where

$$\Delta y_j \equiv y_{j+1}-y_j, \quad \Delta y_{j,av} \equiv (\Delta y_j + \Delta y_{j-1})/2$$
$$\nu_{j,av} \equiv (\nu_j + \nu_{j+1})/2$$

Here, u and v are the x and y components of velocity, ν is total viscosity, U_e is freestream velocity, Re is Reynolds number, and u_j^n denotes $u(n\Delta x, y_j)$, where Δx is constant. The nonuniform grid is obtained from a one-dimensional transformation due to Roberts [5] (see also Anderson et al. [2], p. 247), which seeks to provide a smoothly varying grid which resolves multiple-length scales, including both the sublayer region and overall boundary-layer thickness. The pressure gradient for internal flow is obtained using a secant iteration (Briley [6]; or Anderson et al. [2], p. 385), which gives the exact result after one iteration (three function evaluations) for the present linearized problem. The algebraic turbulence models used assume adequate resolution of the sublayer region and include various combinations for the outer layer (Prandtl mixing length, Clauser or Baldwin/Lomax eddy viscosity) and the inner layer (Spalding's inner-layer model, or van Driest's mixing length). These more or less standard simple algebraic models are covered in Anderson et al. [2], Shetz [3], and White [4].

COMPUTER CODE

A skeletal description of the modular program (GENBL) is given below:

```
PROGRAM GENBL
    [initialize program variables]
    CALL NUGRID [computes nonuniform
    grid (optional)]
    [define starting velocity profile and
    freestream velocity distribution]
    CALL BLPARAM [computes initial
    boundary-layer parameters: δ, δ*,
    θ, Cf]
    CALL OUTPUT [writes initial solu-
    tion data to output file]
    CALL PLOT [writes initial data to
    plot file]

  DO N=2,NMAX [streamwise marching loop]
    CALL TURB1 [computes turbulent vis-
    cosity (if case is turbulent)]
    CALL XSTEP [advances solution one
    streamwise step]
      [define implicit coefficients and
      boundary conditions for x-momentum
      equation]
      CALL TRIDIAG [solves tridiagonal
      system for u]
      [solve continuity equation for v]
    [update solution arrays]
    CALL BLPARAM [computes and saves
    boundary-layer parameters at each
    step]
    [test for flow separation]
  END DO
    CALL OUTPUT [writes final solution
    data to output file]
    CALL PLOT [writes final data to plot
    file]
END
```

An example of the uncompleted code given to students follows:

```
    SUBROUTINE XSTEP (N, JMAX, Y, U, UN,
   1                  V, VN, TVIS)
C
C THIS SUBROUTINE COMPUTES U, V FROM
C UN, VN USING AN IMPLICIT SCHEME
C
    PARAMETER (NXDIM = 1000,
   1           NYDIM = 1000)
    COMMON/COM1/RE, DX, DY, RKAP, YPA
    COMMON/COM2/UE(NXDIM), DUEDX
                (NXDIM)
   1 DIMENSION Y (NYDIM), U (NYDIM),
   2 UN (NYDIM), V (NYDIM), VN (NYDIM),
   3 TVIS (NYDIM)
     DIMENSION A (NYDIM), B (NYDIM),
   1   C (NYDIM), D (NYDIM)
C
C ******************************
C *** MODIFY CODE HERE TO @@@ BELOW ***
C ******************************
C SOLVE X-MOMENTUM EQUATION FOR U
C DEFINE IMPLICIT COEFFICIENTS AND RHS
   DO 20 J = 2, JMAX − 1
C ?????
C A(J) = ?????
C B(J) = ?????
C C(J) = ?????
C D(J) = ?????
20 CONTINUE
C DEFINE IMPLICIT COEFFICIENTS
C FOR BOUNDARY CONDITIONS
C B (1) = ?????
C C (1) = ?????
C D (1) = ?????
C A (JMAX) = ?????
C B (JMAX) = ?????
C D (JMAX) = ?????
C SOLVE TRIDIAGONAL SYSTEM
C (SOLUTION IS RETURNED IN ''A'')
   CALL TRIDIAG (A, B, C, D, 1, JMAX)
   DO 40 J = 1, JMAX
40 U (J) = A (J)
C SOLVE CONTINUITY EQUATION FOR V
   DO 60 J = 2, JMAX
C ?????
C V (J) = ?????
60 CONTINUE
C ******************************
C*** @@@ END OF CODE MODIFICATIONS ***
C ******************************
      RETURN
      END
```

ASSIGNMENTS

The initial assignment follows lectures (2–3 weeks) on the basics of finite-difference/finite-volume schemes and their application to the laminar boundary-layer equations. The second assignment follows lectures (concurrent with assignment 1) on turbulence and turbulence modeling. The third assignment follows lectures (concurrent with assignment 2) on introduction to numerical methods for

the Navier–Stokes equations, including algorithms for model equations.

Assignment 1

Study the code GENBL and complete the missing portions as indicated: Complete SUBROUTINE XSTEP to implement the implicit solution algorithm. Complete SUBROUTINE BLPARAM to compute the boundary-layer parameters. Modify SUBROUTINE PLOT for compatibility with the plotting software used. Compute the following flow cases:

Case 1.1: Laminar Flow Past a Flat Plate. Solve for the region from $x = 0.05$ m to $x = 1.0$ m, using 41 and 100 points for the x and y directions, respectively. Use a Reynolds number of 50,000 based on a reference length of 1.0 m. Locate the outer boundary at $y = 0.05$ m, and use a cubic profile for the initial condition. Plot boundary-layer thickness δ, displacement thickness δ^*, and skin friction coefficient C_f versus x, and compare with the Blasius solution to validate the completed code. Plot the initial and final velocity profiles.

Case 1.2: Laminar Flow Solution for Specified Pressure Gradient. Solve for flow(s) with specified pressure gradient (e,g., Shetz [3], p. 112; Example 4-2: Flow past a flat plate with a ramp).

Case 1.3: Internal Flow in a Two-Dimensional Channel. Modify the code to compute internal flow by using a secant iteration to find the pressure gradient at each step which gives a constant flowrate. Solve for flow in a two-dimensional channel, from $x = 0.05$ m to $x = 50.0$ m, using 200 and 100 points for the x and y directions, respectively. Assume symmetry in the y direction, and take the channel half width to be 0.05 m. Use a Reynolds number of 5000 based on channel width. Take the initial boundary-layer thickness to be 0.005 m, and use a cubic profile for the starting velocity profile. Plot pressure p, pressure gradient dp/dx, and centerline velocity u_{C_L} versus x, and compare with measurements of Goldstein and Kreid [7] to validate the code. Plot u versus y for initial and final x locations. Determine the computed flow rate.

Assignment 2

Study additional code inserts (given in an electronic handout) for turbulent flow initialization and for SUBROUTINE NUGRID, which computes a nonuniform grid distribution to provide good resolution in both the outer portion of the boundary layer and in the sublayer region. These code inserts are to be added to the code completed for case 1.1 in assignment 1.

Modify SUBROUTINE TURB1 to implement the assigned turbulence models. Solve for turbulent flow past a flat plate, from $x = 5.0$ m to $x = 6.0$ m, using 41 points for the x direction. Use a Reynolds number of 10^6 based on a reference length of 1.0 m. Use a Coles profile to define the starting velocity profile, and for convenience, ignore the sublayer region in the starting profile (the sublayer is quickly established during the marching solution). Locate the outer boundary at 2.5 times the initial boundary-layer thickness, which is estimated by assuming the leading edge is at $x = 0$. For the following flow cases, plot u^+ versus y^+ (compare this with the log law and sublayer profiles to validate the code), u versus y, and δ, δ^*, C_f versus x.

Case 2.1: Clauser/Spalding Turbulence Model. Compare computed results for uniform grids using 40, 100, and 500 points in the y direction, to obtain an indication of the accuracy obtainable using equally spaced grids for turbulent flow.

Case 2.2: Clauser/Spalding Turbulence Model. For each given grid and stretching parameter ϵ, compare results for 30 points ($\epsilon = 0.002$) and 200 points ($\epsilon = 0.01$) in the y direction.

Case 2.3: Various Turbulence Models. Using 200 points ($\epsilon = 0.01$), compare the solutions obtained using different turbulence models (e.g., Clauser/Spalding, Prandtl/van Driest, and Baldwin/Lomax/Prandtl/van Driest models).

Assignment 3

Modify the code GENBL or write a new code to solve the nonlinear viscous Burger's equation using an explicit scheme and a linearized implicit scheme. Run assigned cases for different grids and time steps which illustrate concepts such as stability, accuracy, and iterative convergence rate.

TYPICAL RESULTS

Case 1.1: Laminar Flow Past a Flat Plate

The computed results for boundary-layer parameters δ, δ^*, and C_f are in reasonable agreement with

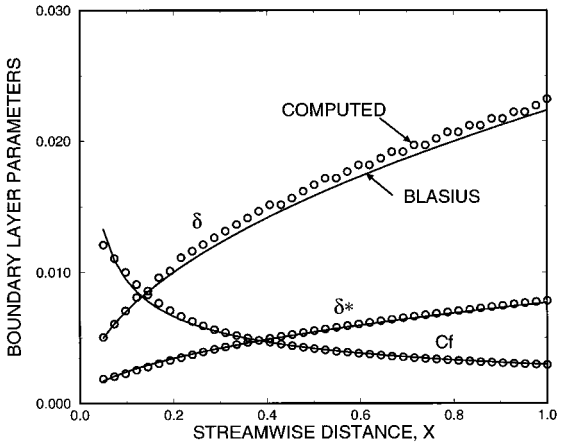

Figure 1 Laminar flow past a flat plate.

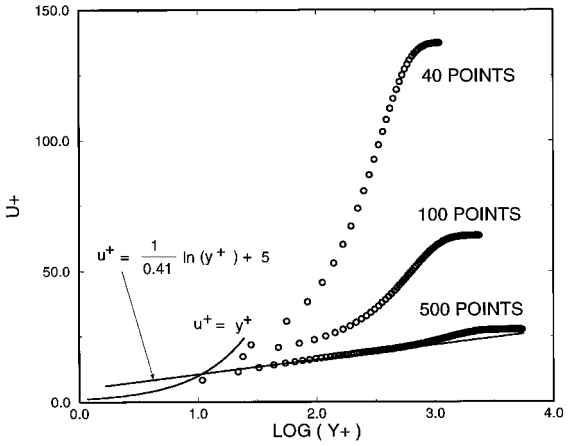

Figure 3 Uniform grid: effect of mesh size (flat plate, Clauser/Spalding turbulence model).

the Blasius solution (Fig. 1). This provides a validation of the code for a case whose solution is known. At this point, the student can easily experiment with different step sizes and mesh spacings to see what effects these have on accuracy.

Case 1.3: Internal Flow in a Two-Dimensional Channel

Computed results are shown for velocity profile at different x locations (Fig. 2). The variables are nondimensional, with average velocity and channel width as the reference quantities.

Case 2.1: Turbulent Flow Past a Flat Plate with Uniform Grid

For the uniform grid, there is inadequate resolution of the sublayer (Fig. 3). Even with 500 points across the boundary layer, the first point adjacent to the wall is located at y^+ of order 10. The point is that even though the numerical method gives results for only 40 points, the solutions are seriously deficient with regard to wall shearing stress (the same would be true for heat transfer).

Case 2.2: Turbulent Flow Past a Flat Plate with Nonuniform Grid

For the nonuniform grid, points are redistributed to provide adequate sublayer resolution (Fig. 4). Here, reasonably good results are obtained with only 30 carefully distributed points. The computed C_f is 0.00278 (30 points) and 0.00283 (200 points).

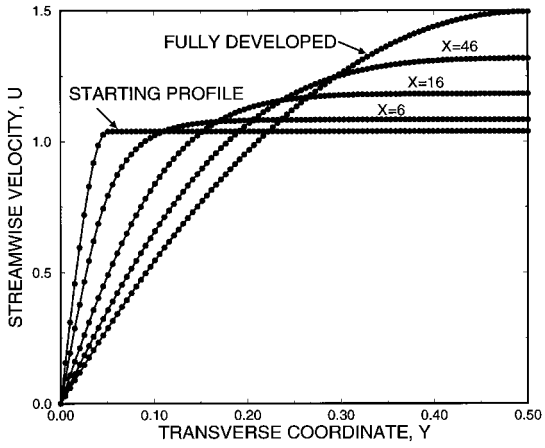

Figure 2 Internal flow in a two-dimensional channel.

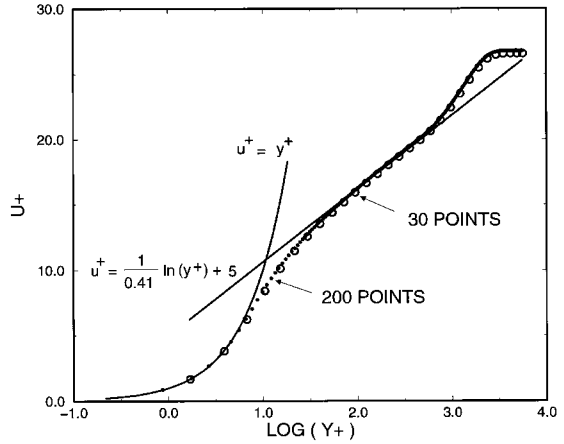

Figure 4 Nonuniform grid: effect of mesh size (flat plate, Clauser/Spalding turbulence model).

STUDENT REACTION

This project has been used twice as of this writing. Starting with the partially completed program after 9 h of lecture, students who were not accomplished programmers were able to complete assignment 1 in about 40 h spread over 4 weeks. One student experienced in programming needed 30–40 h and spent a lot of time understanding the methodology. Another more experienced student needed only 20 h. At this point (midterm), the students were basically up to speed and were able to finish the remaining assignments without too much difficulty. The two most common FORTRAN programming errors were due to (1) not understanding integer arithmetic (i.e., $1/2 \Rightarrow 0$, whereas $1./2. \Rightarrow 0.5$) and (2) array indices out of range at wall or freestream boundaries.

Student reaction has been favorable. Several students said in their course evaluations that the project was their favorite part of the course (there were no complaints). The following responses were given in a follow-up interview with a student who had been inexperienced in programming: The first assignment took at least 30 h. Overall, the project took a lot of time, but it was a very reasonable workload for the course. The project was very helpful in learning how to program and in understanding numerical methods, especially implicit methods. It would have been very difficult to write a completely new program for this project. The most difficult part was understanding how to apply the secant method for internal flow and how to program the Baldwin–Lomax turbulence model. It was informative to plot u^+ versus y^+ for a turbulent boundary layer. A second student who was an accomplished programmer said that the programming and debugging were no problem at all, but understanding *what* to program took time and effort.

CONCLUSIONS

In developing the present project, an effort was made to give students exposure to computational methodology, code development, and case running without having them spend extensive time writing a completely new computer program. In practice, this leaves time for more depth of coverage and case running, such as the solution for internal flow. Since all students work with the same basic code, and since cases are specified in detail, the students should obtain exactly the same results as the instructor. Consequently, the instructor can more easily pinpoint what is going wrong if a student needs help. Since students modify only a small part of the code, it has proven very easy to spot even minor or subtle programming errors. Although experienced students should have no trouble debugging their own codes, this project format allows considerable flexibility in guiding students of differing background and abilities. With a less-controlled format, an inexperienced student may obtain results which are not obviously wrong and never realize the results are incorrect. Overall, this project seems to work well regardless of the students' programming skills.

AVAILABILITY OF INSTRUCTIONAL MATERIALS

Copies of the completed computer program, assignments, and relevant documentation are available from the first author (e-mail: briley@erc.msstate.edu).

REFERENCES

[1] H. Petroski, ''Failed promises,'' *Am. Sci.*, Vol. 82, 1994, pp. 6–9.

[2] D. A. Anderson, J. C. Tannehill, and R. H. Pletcher, *Computational Fluid Mechanics and Heat Transfer*, Hemisphere Publishers, New York, 1984.

[3] J. A. Shetz, *Boundary Layer Analysis*, Prentice Hall, Englewood Cliffs, NJ, 1993.

[4] F. M. White, *Viscous Fluid Flow*, 2nd Ed., McGraw-Hill, New York, 1991.

[5] G. O. Roberts, ''Computational meshes for boundary layer problems,'' in *Proceedings of the Second International Conference on Numerical Methods of Fluid Dynamics, Lecture Notes in Physics*, Vol. 8, Springer-Verlag, New York, 1971, pp. 171–177.

[6] W. R. Briley, ''Numerical method for predicting three-dimensional steady viscous flow in ducts,'' *J. Comp. Phys.*, Vol. 14, 1974, pp. 8–28.

[7] R. J. Goldstein and D. K. Kreid, ''Measurement of laminar flow development in a square duct using a laser-Doppler flowmeter,'' *ASME J. Appl. Mech.*, Vol. 34, 1967, pp. 813–818.

BIOGRAPHIES

W. Roger Briley received his mechanical engineering BS degree from Louisiana Tech University, and his MS and PhD from the University of Texas at Austin. He has worked at United Technologies Research Center and was a cofounder of Scientific Research Associates in Glastonbury, Connecticut. Currently, he is a professor of mechanical engineering and is affiliated with the NSF Engineering Research Center for Computational Field Simulation at Mississippi State University, where he teaches graduate and undergraduate courses in the thermal and fluid sciences. He is actively involved in multidisciplinary research and education activities involving development and application of field simulation algorithms and parallel computing for complex engineering problems.

B. K. Hodge received his aerospace engineering BS and MS degrees from Mississippi State University (MSU) and his mechanical engineering MS and PhD from the University of Alabama. He has worked for Thiokol Corporation and Sverdrup (AEDC). Currently, he is a Giles Distinguished Professor, a Hearin-Hess Professor of Engineering, and a professor of mechanical engineering at MSU, where he teaches graduate and undergraduate courses in the thermal sciences and conducts related research. In 1993 he was named as one of four master teachers at MSU. Since joining the faculty, Dr. Hodge has developed six new courses and written two textbooks. He serves as the undergraduate coordinator for the mechanical engineering program at MSU.