Turbulence Model Implementation and Verification in the SENSEI CFD Code

Charles W. Jackson*
NASA Langley Research Center, Hampton, VA, 23681

William C. Tyson† and Christopher J. Roy‡
Virginia Tech, Blacksburg, VA, 24061

This paper outlines the implementation and verification of the negative Spalart-Allmaras turbulence model into the SENSEI CFD code. The SA-neg turbulence model is implemented in a flexible, object-oriented framework where additional turbulence models can be easily added. In addition to outlining the new turbulence modeling framework in SENSEI, an overview of the other general improvements to SENSEI is provided. The results for four 2D test cases are compared to results from CFL3D and FUN3D to verify that the turbulence models are implemented properly. Several differences in the results from SENSEI, CFL3D, and FUN3D are identified and are attributed to differences in the implementation and discretization order of the boundary conditions as well as the order of discretization of the turbulence model. When a solid surface is located near or intersects an inflow or outflow boundary, higher order boundary conditions should be used to limit their effect on the forces on the surface. When the turbulence equations are discretized using second order spatial accuracy, the edge of the eddy viscosity profile seems to be sharper than when a first order discretization is used. However, the discretization order of the turbulence equation does not have a significant impact on output quantities of interest, such as pressure and viscous drag, for the cases studied.

I. Introduction

This paper summarizes the implementation of turbulence modeling in the research code SENSEI (Structured, Euler/Navier-Stokes Explicit-Implicit solver). SENSEI is a compressible, cell-centered, finite-volume CFD code written in modern Fortran that previously could only solve the Euler and laminar Navier-Stokes equations [1]. For this paper, SENSEI is upgraded to solve the Favre-Averaged Navier-Stokes equations with a linear eddy viscosity turbulence model: the negative variant of the Spalart-Allmaras one-equation model [2, 3]. The governing equations and their discretizations are presented in detail. Since SENSEI is a working research code with multiple developers, a major priority is to implement the turbulence models in the least intrusive way possible while remaining flexible enough to easily incorporate additional models in the future. This is achieved by implementing the turbulence models in an object-oriented manner. The object-oriented approach, which is discussed in detail, improves both the modularity and simplicity of the code base. To further exploit these benefits, the object-oriented paradigm is used to redesign several other areas of the code including the linear system and the boundary conditions. These modifications improve the flexibility and simplicity of the code and have facilitated the implementation of new capabilities.

Verification cases are run to ensure proper implementation of the turbulence models. These are the 2D verification cases provided by the Turbulence Modeling Resource (TMR) website [4] and include flow over a zero pressure gradient flat plate, a coflowing jet, a bump-in-channel, and flow around an airfoil. SENSEI results for these cases are presented along with a comparison with TMR results from CFL3D [5] and FUN3D [6]. It is common for other codes to only use first-order spatial discretizations for the turbulence equations, while using second-order discretizations for the mean flow equations. This is the case for the results from FUN3D and CFL3D presented on the TMR website. Because of how the turbulence models are implemented in SENSEI, it is possible to change the discretization order of the turbulence equation. This design feature is used to examine the effect of using mixed spatial discretization orders on the final solution. This paper also examines the effect of boundary condition implementations on the final solution. Specifically,

*Research Student Trainee, Computational AeroSciences Branch, 15 Langley Blvd. MS 128, AIAA Member.
†Ph.D. Candidate, Kevin T. Crofton Department of Aerospace and Ocean Engineering, 215 Randolph Hall, AIAA Member.
‡Professor, Kevin T. Crofton Department of Aerospace and Ocean Engineering, 215 Randolph Hall, AIAA Associate Fellow.

different order boundary conditions are used and the effect these differences have on quantities of interest are reported for each problem.

II. FANS Governing Equations

The Favre-Averaged Navier-Stokes equations (FANS) describe the motion of a compressible, turbulent fluid and are a statement of conservation of mass, momentum, and energy [7]. They are obtained by performing Favre-averaging on the Navier-Stokes equations. There are some terms that are modeled and some approximations including

- the Boussinesq approximation to model the Reynolds stress tensor,
- an approximation to model the turbulent heat flux,
- and a model for the lumped together molecular diffusion and turbulent transport terms.

These approximations and the derivation of the final form of the FANS equations are presented in detail in Appendix A. The final form of the FANS equations solved is

$$\begin{aligned} \frac{\partial \bar{\rho}}{\partial t} + \frac{\partial}{\partial x_i}\left(\bar{\rho}\tilde{u}_i\right) &= 0, \\ \frac{\partial}{\partial t}\left(\bar{\rho}\tilde{u}_i\right) + \frac{\partial}{\partial x_j}\left[\bar{\rho}\tilde{u}_j\tilde{u}_i + p\delta_{ij} - \tilde{\tau}_{ij}\right] &= 0, \\ \frac{\partial}{\partial t}\left(\bar{\rho}e_t\right) + \frac{\partial}{\partial x_j}\left[\bar{\rho}\tilde{u}_j h_t - k_{eff}\frac{\partial \tilde{T}}{\partial x_j} - \left(\mu + \frac{\mu_T}{\sigma_k}\right)\frac{\partial k}{\partial x_j} - \tilde{u}_i\tilde{\tau}_{ij}\right] &= 0, \end{aligned} \tag{1}$$

where the combination of the laminar and turbulent stresses is defined as

$$\tilde{\tau}_{ij} = 2\mu_{eff}\left(S_{ij} - \frac{1}{3}\frac{\partial \tilde{u}_k}{\partial x_k}\delta_{ij}\right) - \frac{2}{3}\bar{\rho}k\delta_{ij}, \tag{2}$$

and the effective viscosity, μ_{eff}, and effective thermal conductivity, k_{eff}, are defined as

$$\mu_{eff} = \mu + \mu_T, \tag{3}$$

and

$$k_{eff} = C_p\left(\frac{\mu}{Pr_L} + \frac{\mu_T}{Pr_T}\right). \tag{4}$$

Finally, S_{ij} is the mean strain rate, Pr_L is the (laminar) Prandtl number, and Pr_T is the turbulent Prandtl number. To fully close the system, a turbulence model must be used to model the turbulent eddy viscosity, μ_T, and the turbulent kinetic energy, k.

III. Spalart-Allmaras Turbulence Model

A. Governing Equations

The Spalart-Allmaras turbulence model [2] (SA) is a one-equation turbulence model that models the turbulent eddy viscosity, ν_T, using a turbulence working variable, $\tilde{\nu}$. The turbulence working variable, $\tilde{\nu}$, is related to the turbulent eddy viscosity, ν_T, through the relationship

$$\nu_T = \tilde{\nu} f_{v1}, \quad f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3}, \quad \chi = \frac{\tilde{\nu}}{\nu}. \tag{5}$$

The behavior of the turbulence working variable, $\tilde{\nu}$, is governed by the transport equation

$$\frac{\partial}{\partial t}\left(\bar{\rho}\tilde{\nu}\right) + \frac{\partial}{\partial x_j}\left(\bar{\rho}\tilde{u}_j\tilde{\nu}\right) = \bar{\rho}\left(P - D\right) + \frac{1}{\sigma}\frac{\partial}{\partial x_k}\left[\bar{\rho}\left(\nu + \tilde{\nu}\right)\frac{\partial \tilde{\nu}}{\partial x_k}\right] + \frac{c_{b2}}{\sigma}\bar{\rho}\left(\frac{\partial \tilde{\nu}}{\partial x_k}\frac{\partial \tilde{\nu}}{\partial x_k}\right) - \frac{1}{\sigma}\left(\nu + \tilde{\nu}\right)\frac{\partial \bar{\rho}}{\partial x_k}\frac{\partial \tilde{\nu}}{\partial x_k}, \tag{6}$$

where P is a production term, D is a destruction term, and σ and c_{b2} are constants. The production and destruction terms are given by

$$P = c_{b1}\left(1 - f_{t2}\right)\tilde{S}\tilde{\nu}, \quad D = \left(c_{w1}f_w - \frac{c_{b1}}{\kappa^2}f_{t2}\right)\left[\frac{\tilde{\nu}}{d}\right]^2. \tag{7}$$

The modified vorticity, $\tilde{S}$, is given by

$$\tilde{S} = S + \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2}, \quad f_{v2} = 1 - \frac{\chi}{1 + \chi f_{v1}}, \tag{8}$$

where S is the magnitude of vorticity and d is the distance to the nearest wall. The remaining terms are given by

$$f_{t2} = c_{t3} exp\left(-c_{t4}\chi^2\right), \quad f_w = g\left[\frac{1 + c_{w3}^6}{g^6 + c_{w3}^6}\right]^{1/6}, \quad g = r + c_{w2}\left(r^6 - r\right), \quad r = min\left(\frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2}, r_{lim}\right). \tag{9}$$

The empirical constants for this turbulence model are:

$c_{b1} = 0.1355$	$\sigma = 2/3$	$c_{b2} = 0.622$	$\kappa = 0.41$
$c_{w1} = \frac{c_{b1}}{\kappa^2} + \frac{(1 + c_{b2})}{\sigma}$	$c_{w2} = 0.3$	$c_{w3} = 2$	$c_{v1} = 7.1$
$c_{t1} = 1$	$c_{t2} = 2$	$c_{t3} = 1.2$	$c_{t4} = 0.5$
$r_{lim} = 10$			

As Allmaras et al. [3] point out, the original SA model was written in nonconservation form and is valid for both incompressible and compressible flows. However, in order to easily implement the equation into the current finite-volume framework in SENSEI, Eq. 6 is written in conservation form by combining the original equation with conservation of mass. Also, because the Spalart-Allmaras model does not model the behavior of the turbulent kinetic energy, all terms in Eq. 1 containing the turbulent kinetic energy are neglected.

B. Boundary Conditions

The boundary conditions for the turbulence working variable, $\tilde{\nu}$, are given as

$$\text{no-slip wall:} \quad \tilde{\nu} = 0, \qquad \text{symmetry plane:} \quad \frac{\partial \tilde{\nu}}{\partial n} = 0, \qquad \text{freestream:} \quad \frac{\tilde{\nu}}{\nu} \approx 3 \text{ to } 5.^*$$

C. Modifications Implemented

To ensure that the numerical solution is physically realizable, it is sometimes necessary to limit or adjust solution variables during the solution procedure to prevent negative values. The modification given by Allmaras et al. [3] is implemented here to prevent negative values of the modified vorticity, $\tilde{S}$, namely

$$\tilde{S} = \begin{cases} S + \bar{S} & : \bar{S} \geq -c_{v2} S \\ S + \frac{S\left(c_{v2}^2 S + c_{v3}\bar{S}\right)}{(c_{v3} - 2c_{v2})S - \bar{S}} & : \bar{S} < -c_{v2} S \end{cases} \tag{10}$$

where $c_{v2} = 0.7$ and $c_{v3} = 0.9$ and

$$\bar{S} = \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2}. \tag{11}$$

To address some of the convergence issues we observe in the initial transient states, we have also chosen to implement the SA-neg model [3]. When solving some of the cases, as the solution is going through initial transients, the working variable can be driven negative, especially near the edge of a wake. However, a negative value of $\tilde{\nu}$ is not allowed in the original SA model, so the working variable in these cells must be floored to zero. Unfortunately, in the next iteration, the update might attempt to reduce the working variable again, causing a situation where the solution is not allowed to evolve further. To address this issue, the SA-neg model was created to allow $\tilde{\nu}$ to be negative. The SA-neg model is the same as the SA model when $\tilde{\nu}$ is greater than or equal to zero. However, when $\tilde{\nu}$ is negative, μ_T is set to zero and Eq. 6 is modified to be

$$\frac{\partial}{\partial t}\left(\bar{\rho}\tilde{\nu}\right) + \frac{\partial}{\partial x_j}\left(\bar{\rho}\tilde{u}_j\tilde{\nu}\right) = \bar{\rho}\left(P_n - D_n\right) + \frac{1}{\sigma}\frac{\partial}{\partial x_k}\left[\bar{\rho}\left(\nu + f_n\tilde{\nu}\right)\frac{\partial \tilde{\nu}}{\partial x_k}\right] + \frac{c_{b2}}{\sigma}\bar{\rho}\left(\frac{\partial \tilde{\nu}}{\partial x_k}\frac{\partial \tilde{\nu}}{\partial x_k}\right) - \frac{1}{\sigma}\left(\nu + f_n\tilde{\nu}\right)\frac{\partial \bar{\rho}}{\partial x_k}\frac{\partial \tilde{\nu}}{\partial x_k}, \tag{12}$$

*This range of the working variable is associated with a fully turbulent freestream value (see [8] for more details). In this study, a freestream value of $\tilde{\nu} = 3.0\nu$ is used.

where

$$P_n = c_{b1}\left(1 - c_{t3}\right) S\tilde{\nu}, \quad D_n = -c_{w1}\left(\frac{\tilde{\nu}}{d}\right)^2, \quad f_n = \frac{c_{n1} + \chi^3}{c_{n1} - \chi^3}, \tag{13}$$

$$\tag{14}$$

and $c_{n1} = 16$. The final solutions for all of the tested cases only have very small regions of negative $\tilde{\nu}$, typically at the edges of the turbulent region.

IV. Finite-Volume Discretization

The Favre-Averaged Navier-Stokes (FANS) equations are discretized using a second-order, cell-centered, finite-volume discretization. In weak, conservation form, the FANS may be written as

$$\frac{\partial}{\partial t}\int_{\Omega_i} \vec{U}\, d\Omega + \oint_{\partial\Omega_i} \left(\vec{F}_i - \vec{F}_v\right) dS = \int_{\Omega_i} \vec{S}\, d\Omega \tag{15}$$

where $\partial\Omega_i$ is the area of the surface which bounds a fixed control volume, Ω_i, $\hat{n}$ is the outward unit normal on the surface of the control volume, and $\vec{S}$ is a source term. The vector of conserved variables, $\vec{U}$, the inviscid contribution to the flux, $\vec{F}_i$, and the viscous contribution to the flux, $\vec{F}_v$, for the mean flow equations are given by

$$\vec{U} = \begin{bmatrix} \bar{\rho} \\ \bar{\rho}\tilde{u} \\ \bar{\rho}\tilde{v} \\ \bar{\rho}\tilde{w} \\ \bar{\rho}\tilde{e}_t \end{bmatrix}, \qquad \vec{F}_i = \begin{bmatrix} \bar{\rho}\tilde{V}_n \\ \bar{\rho}\tilde{u}\tilde{V}_n + p\hat{n}_x \\ \bar{\rho}\tilde{v}\tilde{V}_n + p\hat{n}_y \\ \bar{\rho}\tilde{w}\tilde{V}_n + p\hat{n}_z \\ \bar{\rho}\tilde{h}_t\tilde{V}_n \end{bmatrix}, \qquad \vec{F}_v = \begin{bmatrix} 0 \\ \tilde{\tau}_{xx}\hat{n}_x + \tilde{\tau}_{xy}\hat{n}_y + \tilde{\tau}_{xz}\hat{n}_z \\ \tilde{\tau}_{xy}\hat{n}_x + \tilde{\tau}_{yy}\hat{n}_y + \tilde{\tau}_{yz}\hat{n}_z \\ \tilde{\tau}_{xz}\hat{n}_x + \tilde{\tau}_{yz}\hat{n}_y + \tilde{\tau}_{zz}\hat{n}_z \\ \tilde{\Theta}_x\hat{n}_x + \tilde{\Theta}_y\hat{n}_y + \tilde{\Theta}_z\hat{n}_z \end{bmatrix}, \tag{16}$$

where $\tilde{V}_n = \tilde{u}\hat{n}_x + \tilde{v}\hat{n}_y + \tilde{w}\hat{n}_z$ is the velocity normal to the surface of the control volume and the components of $\tilde{\Theta}$ are

$$\begin{aligned} \tilde{\Theta}_x &= \tilde{u}\tilde{\tau}_{xx} + \tilde{v}\tilde{\tau}_{xy} + \tilde{w}\tilde{\tau}_{xz} + k_{eff}\frac{\partial\tilde{T}}{\partial x} + MDTT_x, \\ \tilde{\Theta}_y &= \tilde{u}\tilde{\tau}_{xy} + \tilde{v}\tilde{\tau}_{yy} + \tilde{w}\tilde{\tau}_{yz} + k_{eff}\frac{\partial\tilde{T}}{\partial y} + MDTT_y, \\ \tilde{\Theta}_z &= \tilde{u}\tilde{\tau}_{xz} + \tilde{v}\tilde{\tau}_{yz} + \tilde{w}\tilde{\tau}_{zz} + k_{eff}\frac{\partial\tilde{T}}{\partial z} + MDTT_z, \end{aligned} \tag{17}$$

where $MDTT$ is the lumped molecular diffusion and turbulent transport term for the turbulent kinetic energy of a given model and k_{eff} is the effective thermal conductivity given by Eq. 4. The SA-neg model does not use the lumped $MDTT$ term but it is included in other turbulence models such as Menter's k-ω SST model [9].

MUSCL extrapolation [10] is used to reconstruct the left and right primitive states at the faces giving SENSEI a second-order accurate spatial discretization. These left and right states are used to estimate the inviscid fluxes using Roe's flux difference splitting method [11]. The viscous fluxes are computed using a central flux scheme where the analytic flux at an interface is computed using the average of the left and right states. The Green-Gauss theorem is used to compute face gradients for the viscous flux by integrating around a subvolume centered at a given face. The nonlinear solution is marched in time to a steady-state using either explicit Runge-Kutta time integration [12] or the fully implicit backward Euler method as described in [1].

The conserved variable vector, $\vec{U}$, and the inviscid and viscous flux vectors, $\vec{F}_i$ and $\vec{F}_v$, for the Spalart-Allmaras turbulence model are given by

$$\vec{U} = \left[\bar{\rho}\tilde{\nu}\right], \quad \vec{F}_i = \left[\bar{\rho}\tilde{\nu}\tilde{V}_n\right], \quad \vec{F}_v = \left[\tilde{\Upsilon}_x\hat{n}_x + \tilde{\Upsilon}_y\hat{n}_y + \tilde{\Upsilon}_z\hat{n}_z\right], \tag{18}$$

where

$$\tilde{\Upsilon}_x = \frac{\bar{\rho}}{\sigma}\left(\nu + \tilde{\nu}\right)\frac{\partial\tilde{\nu}}{\partial x}, \quad \tilde{\Upsilon}_y = \frac{\bar{\rho}}{\sigma}\left(\nu + \tilde{\nu}\right)\frac{\partial\tilde{\nu}}{\partial y}, \quad \tilde{\Upsilon}_z = \frac{\bar{\rho}}{\sigma}\left(\nu + \tilde{\nu}\right)\frac{\partial\tilde{\nu}}{\partial z}. \tag{19}$$

The source term for the Spalart-Allmaras turbulence model is given by

$$\vec{S} = \left[\bar{\rho}\left(P - D\right) + \frac{c_{b2}}{\sigma}\bar{\rho}\left[\left(\frac{\partial\tilde{\nu}}{\partial x}\right)^2 + \left(\frac{\partial\tilde{\nu}}{\partial y}\right)^2 + \left(\frac{\partial\tilde{\nu}}{\partial z}\right)^2\right] - \frac{1}{\sigma}\left(\nu + \tilde{\nu}\right)\left[\frac{\partial\bar{\rho}}{\partial x}\frac{\partial\tilde{\nu}}{\partial x} + \frac{\partial\bar{\rho}}{\partial y}\frac{\partial\tilde{\nu}}{\partial y} + \frac{\partial\bar{\rho}}{\partial z}\frac{\partial\tilde{\nu}}{\partial z}\right]\right]. \tag{20}$$

The modified equation when $\tilde{\nu}$ is negative is discretized in a similar manner.

An important detail in the implementation of turbulence modeling is how to properly treat the inviscid contribution to the turbulent flux. Commonly the turbulent flux is computed in an uncoupled fashion, which treats the turbulence equation as a scalar transport equation. However, many authors have pointed out the fault with this type of treatment [13, 14]. In treating the turbulence equation as a scalar transport equation, the turbulence convects at a velocity, which does not account for the effect of the turbulence. In the implementation used here, users are given the option to compute the inviscid turbulent flux using either the flux function of Roe or the flux function of Roe and Pike [15]. The flux function of Roe treats the flux in the typical uncoupled manner while the flux function of Roe and Pike properly accounts for the effect of the turbulence on the convection velocity. The reader is referred to [14, 15] for more details regarding the proper treatment of the turbulent flux.

V. Improvements in SENSEI 2.0

In addition to the implementation of turbulence modeling, several other modifications have been made to the code base to improve its flexibility, modularity, and speed. These improvements build on the original design choices made in [1], but extend them to include a more object-oriented approach. Several improvements are also made to the implicit solver, enabling much higher CFL values to be used and for the nonlinear system to converge significantly faster.

A. Object Oriented Programming in SENSEI

For the implementation of the turbulence modeling framework, two main goals are established: modularity and encapsulation. Modularity is important because we wanted to be able to add new turbulence models easily in the future using the existing framework. Encapsulation is important because we want the rest of the code to be agnostic as to whether a turbulence model is being used and we want the turbulence model's variables and routines to be protected from the rest of the code base. These two goals are met by using object-oriented programming (OOP).

OOP meets the goal of modularity through polymorphism. Polymorphism allows a user to define a common interface for a routine and have that routine be implemented in multiple ways depending on the type. This means that the majority of the code knows it is using a turbulence model but it does not know the specific turbulence model it is using. In Fortran 03/08, polymorphism is achieved by defining an abstract derived type. The abstract derived type can have data and routines associated with that type. However, because the type is abstract, some of these routines are allowed to remain unimplemented and their implementation is deferred to a child type. This allows the abstract type to have routines that are common to all such objects, such as output and common initialization and destruction. However, the abstract type does define interfaces for these deferred routines. These interfaces allow the rest of the code to use the object in the same manner no matter which specific type is being used. When this abstract type is extended by another derived type, that child type must implement all of the deferred procedures.

Writing code in this object-oriented manner also provides some level of encapsulation. Encapsulation forces separation between different portions of the code through abstraction. This means that one part of the code only has access to another part of the code through a defined (typically polymorphic) interface. This encapsulation protects, for instance, the constants associated with the SA-neg model from being accidentally changed somewhere else in the code because these variables are not accessible outside of the SA-neg model's routines. Going the other way, the turbulence model does not need to know how it is going to be used (for instance in a primal solve, an error estimation procedure, or an adjoint solve). This removes the burden on the programmer to understand the whole program all at once, and allows them to focus on the current portion of the code. Unfortunately, the Fortran 2008 standard requires all deferred procedures to be public if they are being implemented in a different module, reducing the amount of encapsulation compared to C++ or other object-oriented programming languages.[†] To our knowledge, there is not a keyword analogous to the `protected` keyword in C++, which allows classes that extend the parent class to have access to the private routines of the parent. Because of this, Fortran programmers are forced to make public routines that should be private or protected. This will allow routines that should be protected to be called from anywhere, defeating the purpose of

[†]Some versions of some compilers will still allow this behavior but it is not recommended since it does not follow the standard (see interpretation F08/0052 in WG5 document N1875).

encapsulation. Despite this limitation, the object-oriented additions to the Fortran standard allow for encapsulation of some routines and private variables, which is beneficial.

There are also several indirect benefits of using OOP, such as forcing programmers to adopt good programming practices and improving the testability of the code. OOP forces the programmer to plan ahead and decide what routines are needed to achieve a task and how the rest of the code should interact with the new feature. This planning is important in any programming effort, but it is our experience that programming in an object-oriented manner forces the programmer to follow this best practice. Writing a code base in an object-oriented manner can also simplify the software maintenance. Typically, the objects are single entities that are easier to unit test because the derived types can only be interacted with in certain predefined ways, through their interface. As long as those implemented routines are sufficiently tested, the derived type should be safe to use anywhere in the code because you have limited the possibility for unintended side effects. The abstract derived types also assist in software maintenance by reducing repeated code. Routines that all of the types share, such as output, only need to be written once and they are available for all of the extended types. Because the objects are encapsulated, they are easier to use in multiple places in the code and even in other code bases. Because of these inherent benefits, the turbulence modeling and several other parts of the code are now written in an object-oriented manner. These changes are summarized in the following sections.

1. Turbulence Modeling

The authors' approach to implementing the turbulence modeling framework is to create an abstract parent derived type, `turb_model_t`, which contains routines common to all turbulence models such as outputting data to file and initialization. The parent derived type defines the interface for deferred procedures that all turbulence models require but would implement differently. Examples of these deferred procedures are calculating the turbulent eddy viscosity, μ_T, or determining the production, destruction, and dissipation terms in the turbulence equations. For the full list of procedures in `turb_model_t`, see Appendix B. Each turbulence model is then implemented by extending `turb_model_t` and providing implementations to the routines that were deferred by this parent type. This parent-child structure ensures that a minimal amount of code is repeated and that each turbulence model has the flexibility to implement its equations accordingly.

During code start-up and initialization, SENSEI reads in whether a turbulence model is required. SENSEI then creates an instance of the parent turbulence model object with the specific child type corresponding to the desired turbulence model. Because the SA-neg model has been implemented in conservation form using this object-oriented approach, the same routines used to form the residual and LHS for the mean flow are used to form the residual and LHS for the turbulence equations. If no turbulence model is desired for a given computation, a "null" turbulence model child type is used. This "null" turbulence model returns nothing from its routines and is only present so that the interface with the rest of SENSEI is the same whether a turbulence model is being used or not. With this object-oriented structure, it is straight-forward to implement a new turbulence model (even one with multiple equations) since one only has to create a child derived type that implements the deferred procedures and the rest of the code works as expected. For example, the explicit routines for Menter's k-ω SST model [9] only took one day to implement once the framework was complete. The linearizations of the k-ω model have not yet been implemented, which is why this model does not appear in this paper.

2. Linear System

Another area of the code that has been switched to an object-oriented paradigm is the linear system routines. Previously, the data and routines associated with the linear system were stored in several different modules in the code base. While this approach worked, it was not very flexible and was hard coded to use the compressed sparse row (CSR) format with ILU0 preconditioning. These routines have been grouped into logical units and encapsulated in abstract types. The first of these abstract types is the matrix type, which stores a matrix and provides interfaces for routines like matrix-vector products, row scaling, transpose, etc. We then extend the abstract matrix type to implement different matrix storage types such as: dense, COO (coordinate format), CSR, DIA (diagonal format), etc. Likewise, we have encapsulated all of our preconditioners into derived types that can then be applied to any matrix type. We created an iterative solver derived type, which is extended by the different solvers: GMRES [16] and Bi-CGSTAB [17]. Finally, we created a master linear system derived type, which holds an instance of a matrix, a RHS vector, an instance of a preconditioner, and an instance of an iterative solver to use. Because all of the variables and routines associated with the linear system are in one place, it is easy to keep track of all of the parts of the linear solve process. Also, because all of these routines are written in this modular, object-oriented manner, one could use any combination of matrix storage type,

preconditioner type, and iterative solver together (although certain combinations will be more efficient than others).

3. Boundary Conditions

The boundary conditions have also been rewritten in an object-oriented manner. The abstract boundary type stores the cells over which a given boundary condition is applied, which block face the boundary is on for indexing purposes (for instance ξ_{min} or η_{max}), and other useful information. There are also some common initialization routines stored in the abstract boundary derived type. Each different boundary condition (farfield, wall, symmetry, etc.) is then a derived type that extends this boundary type. These boundary condition child derived types must implement the routines to fill ghost cells, calculate ghost cell Jacobians, and, if they are flux-based boundary conditions, calculate the boundary flux and the flux Jacobian. Each boundary condition can also store extra information necessary for that specific boundary condition such as freestream values. Having this common interface greatly reduces the burden on the other parts of the code. The rest of the code no longer has to hold information about where boundaries are, have individual loops over the 6 different block faces, or have switch statements to choose the specific BC. The rest of the code also does not have to know how or where to store the various inputs for the different boundary conditions as each stores the specific information it needs. The polymorphism hides these switch statements so most of the boundary routines have the same form: a loop over each boundary condition and then a loop over the faces associated with that boundary condition. The build-up of the residual Jacobian is also simplified because all boundary conditions have the same interface, which returns the same expected information, despite the fact that each boundary condition has a different Jacobian with respect to different interior stencils.

B. Implicit Solver

SENSEI uses a backward Euler time integration scheme to perform implicit time-stepping. To populate the left hand side (LHS) of this system, the Jacobians of the residual are hand-coded. Because of the complexity of hand-coding the Jacobians, the mean flow governing equations are solved in one linear system and the turbulence equations are solved in another. This means that the Jacobian of the mean flow equations only has to be taken with respect to the mean flow primitive variables. Likewise, the turbulence equations only have to be linearized with respect to the turbulence variables. This explicit coupling between the sets of equations will decrease the stability and convergence rate for these problems [18]. However, the equations are much simpler to linearize if they are linearized as two decoupled systems, and SENSEI is able to converge all of the problems tested without issue.

Nondimensionalizing the governing equations gave large improvements to the performance of our implicit time-stepping. Reference quantities for the density, temperature, and speed of sound are used to nondimensionalize the mean flow equations. Then, based off of the reference temperature and density, a reference viscosity is calculated to nondimensionalize $\tilde{\nu}$. For more details on the nondimensionalization, refer to [19]. Because these equations have been nondimensionalized the condition number for the LHS is greatly improved. For example, the first-order residual Jacobian about the solution to the flat plate problem on the coarsest grid is compared in dimensional and nondimensional form. The condition number for the dimensional mean flow Jacobian is 1.5E+14 while the condition number for the nondimensional Jacobian is 2.8E+06. This is because, after nondimensionalization, all of the equation's residuals are similar in magnitude as opposed to very different orders of magnitude for the dimensional equations. Because the condition number is so much lower, much higher CFL values can be used to solve the nondimensional system without significant loss of precision. These factors allow the nondimensional problem to converge much faster than the dimensional problem.

After the turbulence equations are linearized, it is important to check that all the linearizations are correct and that the build up of the LHS is correct. This is checked through a finite-difference approximation of the LHS. The final solution for the airfoil and flat plate cases are perturbed, the residual recalculated, and a second-order central finite-difference is used to estimate the corresponding term in the LHS. Several issues were found using this method, specifically some errors in the linearizations of the boundary conditions and the gradients. For both of these cases, the implicit LHS matched the finite-difference LHS to within discretization error (1.0E-10). With the correct, nondimensionalized, LHS matrices for the mean and turbulence equations, we were able to converge all of the test cases to machine zero well within 1000 iterations, even on the finest grids of approximately one million cells.

VI. Results

To verify the turbulence model implementation within SENSEI, verification cases are performed and the results are compared to those of CFL3D and FUN3D. These codes are selected for comparison because they have undergone rigorous code verification using MMS for the SA model [20] and results for these codes are publicly available. The verification cases examined are from the NASA Langley Turbulence Modeling Resource (TMR) website [4]. This is an excellent resource describing the models as well as providing both verification and validation cases and results from CFL3D and FUN3D. For this study, all four of the 2D verification cases provided on the TMR website are used:

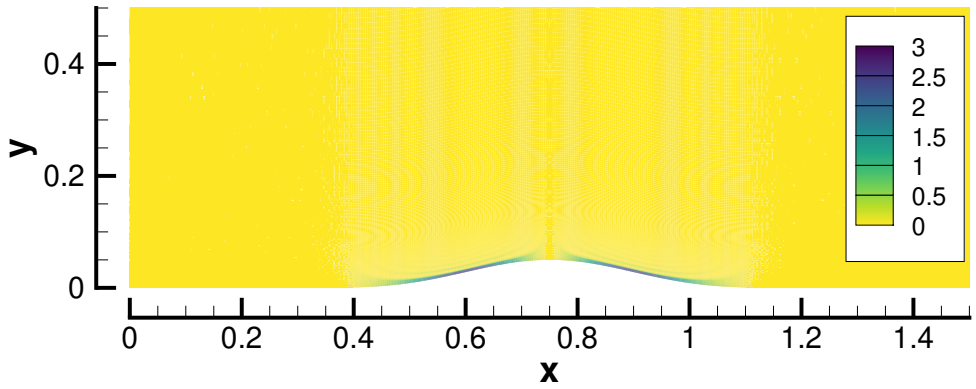
- the 2D zero pressure gradient flat plate (2DZP),[‡]
- the 2D coflowing jet (2DCJ),[§]
- the 2D bump-in-channel (2DB),[¶] and
- the 2D airfoil near-wake (2DANW).[‖]

The airfoil case uses the set of grids from the website that extend the farfield boundary to 500 chord lengths away. For a more detailed description of these cases, including the geometry, grids, and flow state, refer to the TMR website. All of the cases run in SENSEI are converged to machine zero. During the comparison of our implementation of the SA-neg model with other codes, namely CFL3D and FUN3D, several differences are noticed. These differences are determined to stem from using different order discretizations and boundary conditions as well as an error in our wall distance calculation.

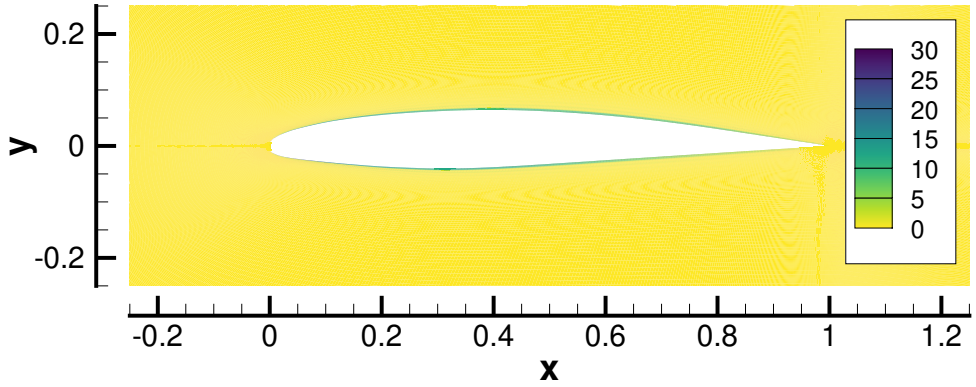
A. Wall Distance

As a first pass, a naive implementation of the wall distance function was implemented to find the distance to the nearest node or face center. While this is easy to do and relatively accurate, it is not exactly correct and even small errors in this distance can have significant effects on the final solution. The correct distance should be the shortest distance to any point on the discrete boundary rather than the small subset of nodes and face centers. The results for the stretched Cartesian meshes (2DZP and 2DCJ) do not change because the wall distance calculated using both the naive and correct methods are the same (the center of the face is the nearest point to the cell centers). However, Fig. 1 shows the calculated wall distance is different for more complex curved boundaries like in the 2DB and 2DANW cases. Fig. 1 shows the relative error in the naive wall distance on the finest grid level for these two cases. The grid lines coming off the bump case are close to normal (as shown in Fig. 2a) so only a small error of at most 3% is committed using the naive approach. However, for the 2DANW case, the grid lines do not come off the airfoil in the normal direction (as shown in Fig. 2b) so some of the naive wall distances are as much as 30% off in the high aspect ratio cells in the boundary layer. These errors, even the small 3% error for 2DB, have a significant effect on the working variable's production and destruction, and thus the final solution. The difference in the skin friction along the bump and airfoil using the two different wall functions can be seen in Fig. 3. It is clear that the regions where the wall distance function is too large are the same regions where the skin friction is also larger than it should be. This makes intuitive sense because the model thinks these cells are farther from the wall then they actually are, so the eddy viscosity in this region is excessively large. The remainder of the results presented in this paper use the correct wall distance calculation.

[‡] https://turbmodels.larc.nasa.gov/flatplate.html
[§] https://turbmodels.larc.nasa.gov/shear.html
[¶] https://turbmodels.larc.nasa.gov/bump.html
[‖] https://turbmodels.larc.nasa.gov/airfoilwakeverif500c.html

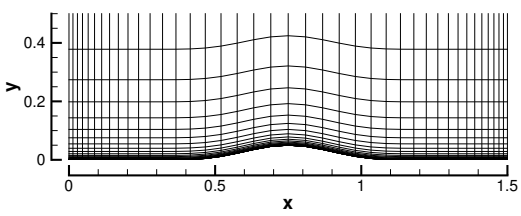


(a) 2DB

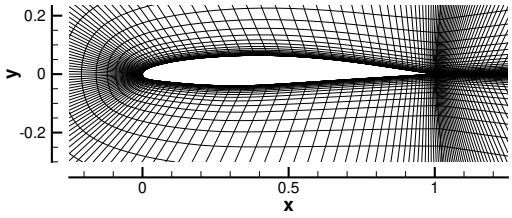


(b) 2DANW

Fig. 1 Percent error in the naive wall distance calculation for 2DB and 2DANW on the finest grid.

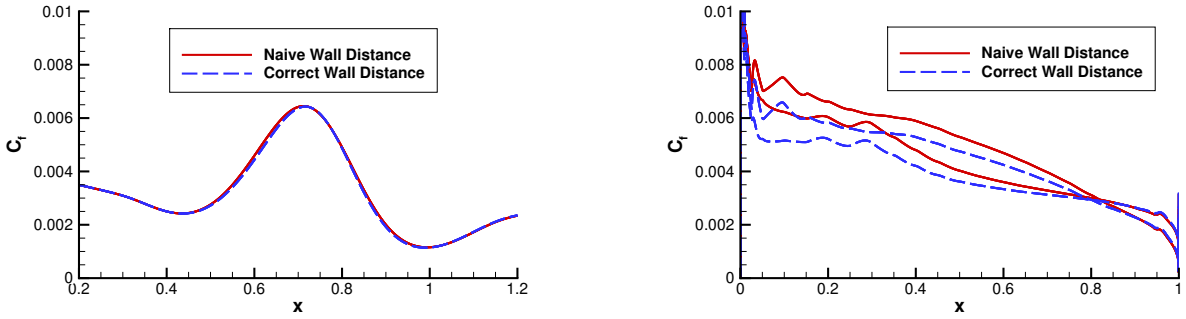


(a) 2DB



(b) 2DANW

Fig. 2 Grid near the surface (at the second coarsest grid level for clarity) for the 2DB and 2DANW.



(a) 2DB **(b) 2DANW**

Fig. 3 Coefficient of friction along the surface calculated using the naive and correct wall distance.

B. Comparison of Codes

To verify that our implementation of the SA-neg model is correct, it is compared to the results provided on the TMR website from, FUN3D and CFL3D. It is a common technique, including the technique employed by FUN3D and CFL3D for the TMR results, to solve the mean flow equations second-order accurate and the turbulence equation first-order accurate (mixed-order). To examine the impact of this decision, SENSEI will be used to solve all of the equations with both second-order and mixed-order discretizations.

Another potential difference between the codes is the boundary condition implementation and order of accuracy. The wall boundary condition in SENSEI is implemented by setting the inviscid flux on the boundary to the pressure flux (since $\tilde{V}_n = 0$ at the wall):

$$\vec{F}_{i,w} = \begin{bmatrix} 0 & p_w \hat{n}_x & p_w \hat{n}_y & p_w \hat{n}_z & 0 \end{bmatrix}^T, \tag{21}$$

where p_w is the pressure at the wall boundary face. The user can select the order of extrapolation used to calculate the pressure at the wall by selecting the order of accuracy to use for the boundary condition. Typically, this would be first-order (constant extrapolation, $p_w = p_1$) or second-order (linear extrapolation, $p_w = 1.5p_1 - 0.5p_2$) where p_1 is the pressure in the cell next to the boundary and p_2 is the pressure in the next interior cell. When integrating surface forces (viscous and pressure forces), SENSEI will use this same extrapolation order to extrapolate the temperature and pressure to the boundary so that the integrated forces match the state used for the discrete boundary condition. When solving the problem with first-order boundary conditions, the final force distributions and integrated quantities do not change significantly if a linearly extrapolated pressure to the wall is used instead of p_w. In most cases, the order of boundary condition should be the same as the interior discretization scheme. However, for some cases, it is necessary to run with lower-order boundary conditions to avoid potential stability issues. Thus, the effect of first- and second-order boundary conditions (denoted as bc1 and bc2) on the final solution is presented below.

1. 2DZP

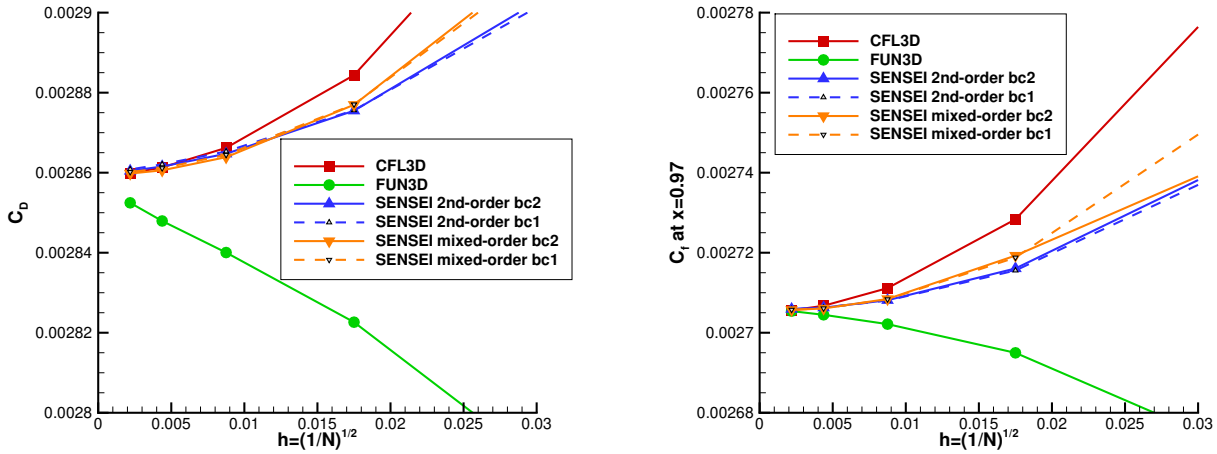
The first case examined is zero pressure gradient flow over a flat plate (2DZP). The grid convergence for the drag coefficient on the plate and the coefficient of friction at $x = 0.97$ are compared in Fig. 4. The observed order of accuracy calculated on the final three grids is shown in Table 1. All of these results match well, with there only being a slight difference between the first- and second-order turbulence on the coarser grids. It is also noted that these quantities are effectively the same regardless of the order of boundary conditions used. However, upon closer examination, some interesting differences became apparent. The differences in the skin friction distribution exist mainly at the leading and trailing edge of the plate as shown in Fig. 5.

Fig. 5a shows that CFL3D and SENSEI, both cell-centered, finite-volume codes, have a much higher skin friction near the leading edge of the plate compared to the node-centered, finite-volume code FUN3D. Fig. 5a also shows that changing the boundary condition order makes essentially no difference in the coefficient of friction at the front of the plate (less than 1.0E-4% difference). However, the order of boundary conditions does have an effect on the skin friction at the trailing edge of the plate, as shown in Fig. 5b.

The end of the plate occurs at the outflow boundary where the no-slip wall intersects the subsonic pressure outflow boundary condition. At this intersection of boundary conditions, the order of extrapolation from the interior to the

ghost cells of the outflow boundary does have an impact on the skin friction (0.3% difference). Fig. 5b shows that if the second-order boundary conditions are used then the skin friction reduces monotonically toward the outflow boundary as expected. However, if a first-order boundary condition is used, then the low-order boundary condition will affect the behavior of the skin friction in the cell at that boundary as well as a significant number of cells upstream (in this case approximately 15 cells). This effect can be seen in both SENSEI and CFL3D, which match very well when using the same settings (mixed-order with first-order outflow).

Changing the discretization order for the turbulence model has a slight effect (approximately 0.03% difference). There is a very slight difference in the velocity profiles for the first and second-order turbulence solution but it is so small that it cannot be seen in a plot. However, one difference that is more significant is the effect on the eddy viscosity profile as shown in Fig. 6. The profile of the normalized turbulence eddy viscosity appears largely the same between the codes, except for the outer edge of the turbulent region. It appears like the edge of the turbulent region is slightly sharper using the higher-order discretization. This is likely because there is less numerical dissipation smearing out the edge of the region of high eddy viscosity. However, this is a fairly small difference away from the body and it does not change the mean flow solution or the quantity of interest (drag) in any significant way on the finer meshes.

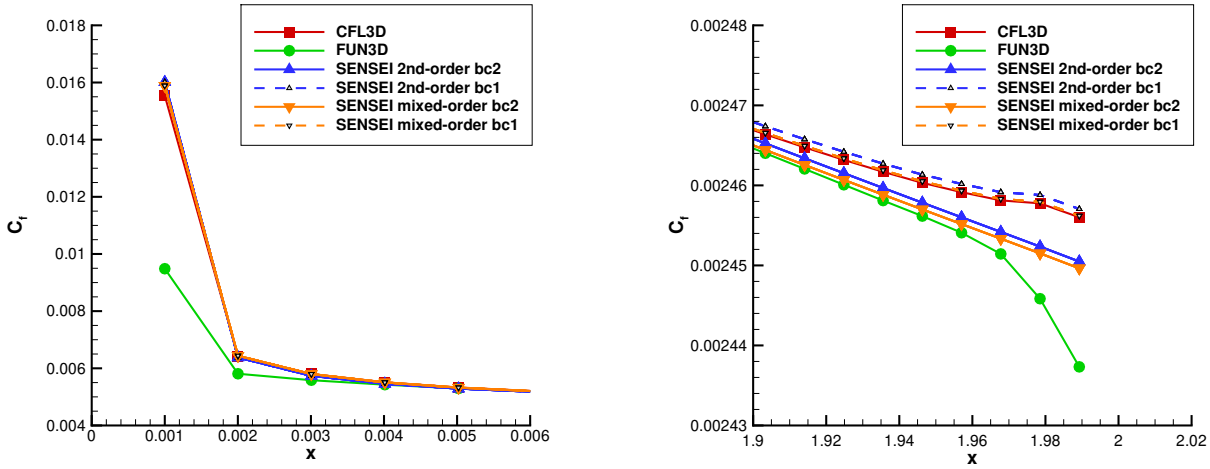


(a) Convergence of C_D. **(b) Convergence of C_f at $x = 0.97$.**

Fig. 4 2DZP: Grid convergence of various quantities.

Table 1 2DZP: Observed order of accuracy on the three finest grids.

	SENSEI second-order	SENSEI mixed-order	CFL3D	FUN3D
Observed Order, C_f	2.26	2.66	1.98	1.34
Observed Order, C_D	1.53	2.04	1.75	0.80



(a) Skin friction at leading edge. **(b) Skin friction at trailing edge.**

Fig. 5 2DZP: Skin friction differences between the different codes and boundary condition orders, highlighting the two regions of difference.

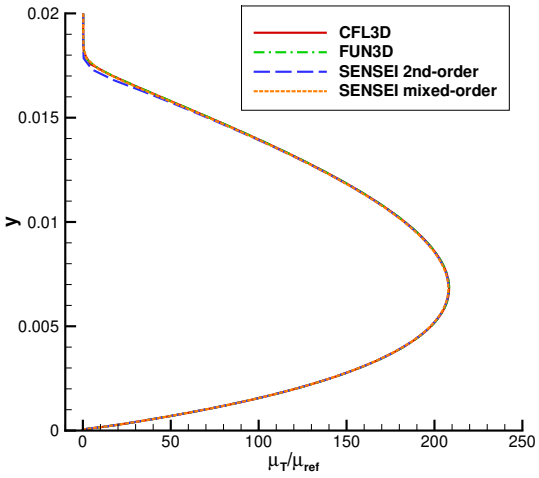


Fig. 6 2DZP: Turbulent eddy viscosity profile at $x = 0.97$.

2. 2DCJ

The next case studied is the coflowing jet case (2DCJ). This case, depicted in Fig. 7, is a fast inner jet and a slower outer freestream flow initially separated by a no-slip wall and then mixing in a shear layer downstream of the dividing plate. For 2DCJ, there are two main quantities of interest used to compare the results of SENSEI to CFL3D and FUN3D: the drag coefficient on the dividing plate, and the local u-velocity at 3 stations. The stations are located on the symmetry plane, $y = 0.0$, and have x coordinates of (1) $x = 2.71623$, (2) $x = 29.2468$, and (3) $x = 95.501$. The grid convergence for these values are presented in Fig. 8, and the observed order of accuracy is provided in Table 2. Highlighting the need to use lower-order boundary conditions for some cases, the solution on the second coarsest grid can not be found when the second-order boundary conditions are used. On this grid level, oscillations in the turbulence working variable set up and never went away, halting the convergence of the problem. These oscillations occurred where the edge of the shear layer meets the outflow boundary. Other than this grid level, all of the grid levels are solved in SENSEI with different

order boundary conditions and the drag coefficient is calculated for those settings. Fig. 8a shows that the drag on the dividing plate is much lower when a first-order boundary condition is used. The boundary condition order has a larger effect on the integrated drag than it did for 2DZP because, in this case, the plate intersects a stagnation temperature and pressure inflow boundary on its leading edge. Because the plate begins at the inflow boundary, errors here will affect the entire plate downstream (not just the end as in 2DZP). Because the inflow boundary affects the entire plate, it will have a larger effect on the integrated quantities of interest (e.g., drag). While it appears that the results with the first-order boundary conditions will eventually converge to the same solution as the other cases, it will likely require a much finer grid.

The turbulence eddy viscosity profile across the jet at station 2 looks largely the same between SENSEI, CFL3D and FUN3D except at the center and edge of the jet, as shown in Fig. 9. The maximum μ_t in the shear layer for the second-order SENSEI solution is lower than the maximum value in the mixed-order solution by 0.21%. Also similar to the 2DZP case, at the edge of the shear layer the eddy viscosity seems to be sharper for the second-order solution rather than more spread out like in the mixed-order solution. These differences are attributed to the fact that SENSEI is solving the turbulence equations second-order accurate, because the mixed-order solution from SENSEI matches the results from FUN3D and CFL3D much more closely (less than 0.001% difference in maximum μ_T). Despite these different values of μ_T, the velocity profiles at this station are nearly identical and the centerline velocity profile is very close except directly downstream of the plate as shown in Fig. 10.

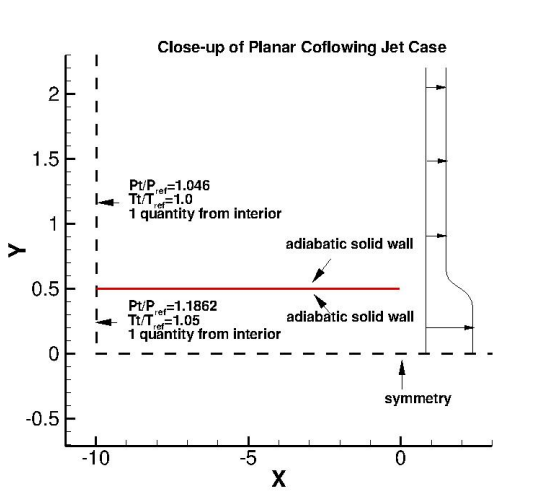
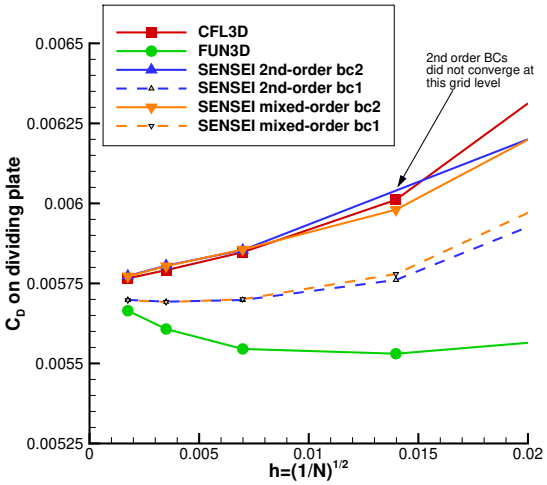
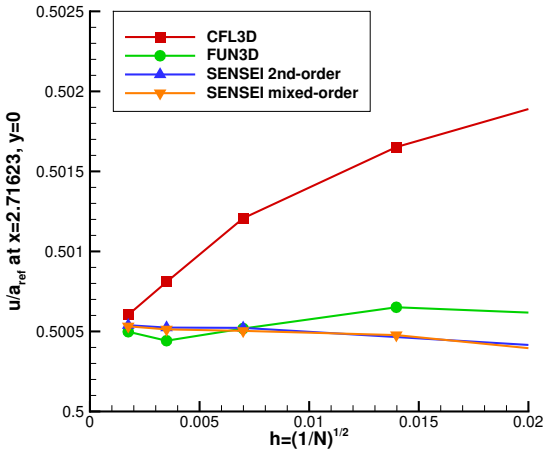


Fig. 7 2DCJ: Geometry and boundary conditions, reproduced from TMR website.[**]

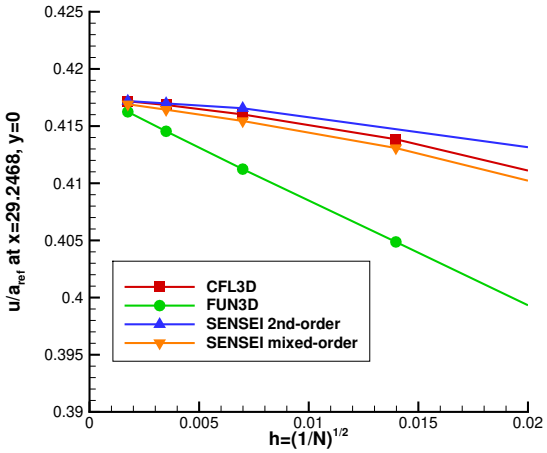
[**]https://turbmodels.larc.nasa.gov/shear.html



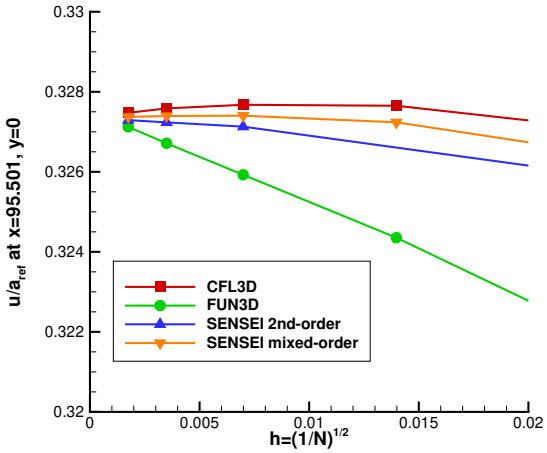
(a) Convergence of C_D.



(b) Convergence of u/a_{ref} at station 1.



(c) Convergence of u/a_{ref} at station 2.

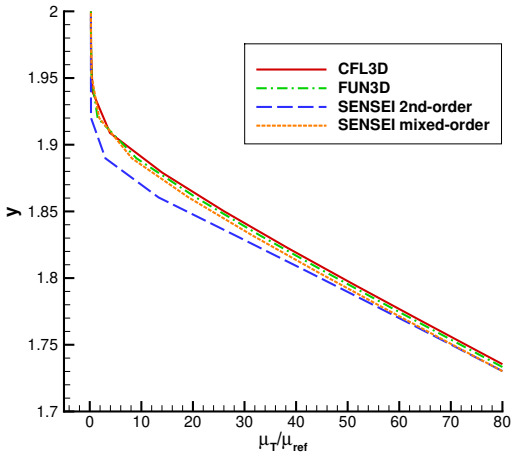


(d) Convergence of u/a_{ref} at station 3.

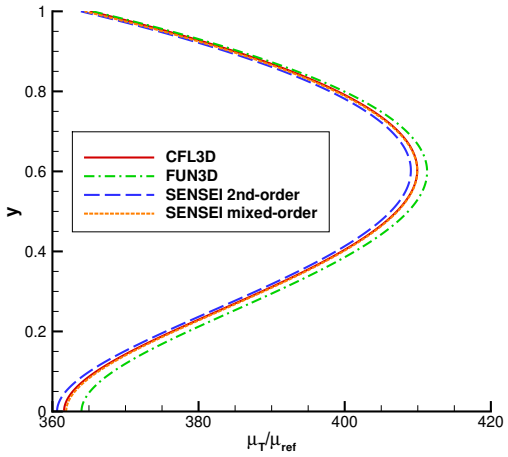
Fig. 8 Grid convergence of various quantities for the 2DCJ.

Table 2 2DCJ: Observed order of accuracy on the three finest grids.

	SENSEI second-order	SENSEI mixed-order	CFL3D	FUN3D
Observed Order, C_D	0.59	0.71	1.13	0.12
Observed Order, u/a_{ref} at station 1	3.16	negative	0.96	oscillatory
Observed Order, u/a_{ref} at station 2	0.99	1.05	1.35	0.96
Observed Order, u/a_{ref} at station 3	0.93	negative	0.29	0.94

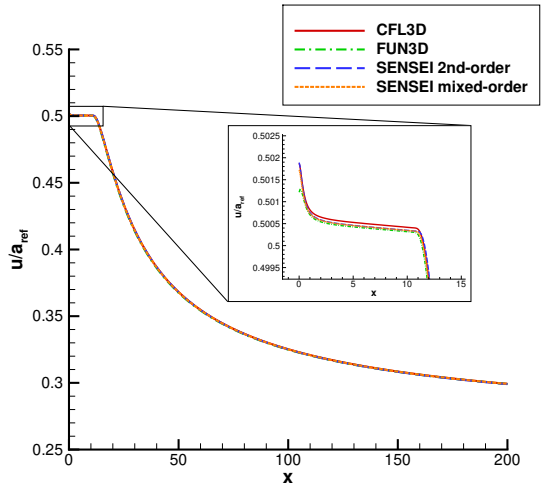


(a) Turbulent eddy viscosity near edge of jet. Note that the second-order turbulent eddy viscosity is narrower than the mixed-order results.

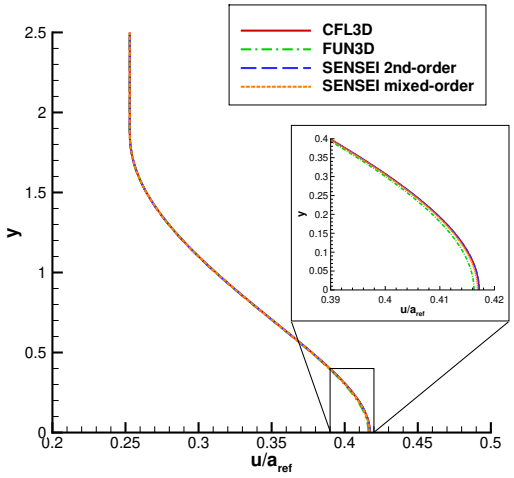


(b) Turbulent eddy viscosity near center of jet. Note that the second-order peak is smaller than the mixed-order results.

Fig. 9 2DCJ: Profile of turbulent eddy viscosity at station 2, highlighting the differences between the codes.



(a) Velocity profile along the symmetry plane.



(b) Velocity profile at station 2.

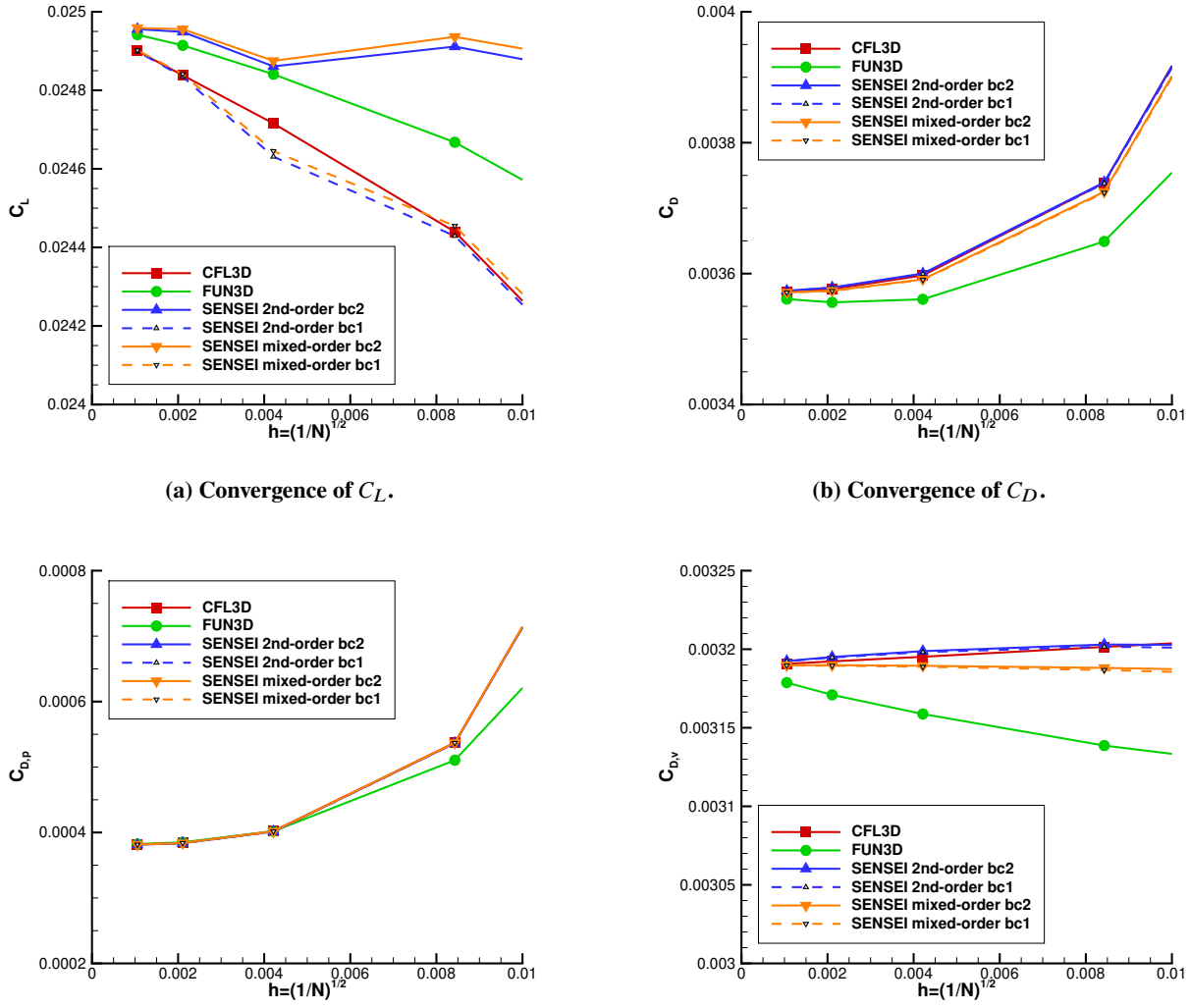
Fig. 10 2DCJ: Various velocity profiles. Note how similar all of these results are.

3. 2DB

The 2D bump case (2DB) has many quantities of interest including the coefficients of lift, pressure drag, viscous drag, and total drag. The grid convergence properties of these values are shown in Fig 11 and the observed order of accuracy is shown in Table 3. The order of boundary conditions has an effect (0.3% to 5% difference) on the lift on the bump, but it does not have an impact on the drag (less than 0.1% difference). This is likely because the order of the boundary condition seems to have a significant effect on the pressure but a lesser effect on the turbulence equation and velocity gradients at the wall. This also explains why the boundary condition order does not make a noticeable difference in the drag on the flat plate because there is only viscous drag in that case. There are no discernible differences between the C_f or C_p distributions for the three codes on the finest grids except for the C_f peaks at the beginning and end of

bump, so they are not presented here.

Similar to the other cases, the order of the turbulence model does not have a significant effect on the integrated quantities of interest. However, changing the spatial discretization order does show differences in the width and magnitude of the turbulent region. Fig. 12, shows the normalized eddy viscosity through the boundary layer at $x = 0.75$, which is the top of the bump. Similar to the flat plate and coflow problems, the edge of the turbulent region in the second-order solution appears to be sharper compared to the mixed-order solution. At this station along the bump, the second-order eddy viscosity peak is approximately 0.5% larger than the mixed-order peak. Fig. 13 shows the maximum value of the the eddy viscosity in the boundary layer along the bump for CFL3D and SENSEI with different orders. This figure shows that for the majority of the problem the maximum eddy viscosity is similar with mixed- and second-order solutions, however, near the top of the bump the second-order eddy viscosity is higher. Again, when SENSEI runs with mixed-order spatial discretizations, the results match very well with CFL3D.



(a) Convergence of C_L.

(b) Convergence of C_D.

(c) Convergence of $C_{D,p}$.

(d) Convergence of $C_{D,v}$.

Fig. 11 2DB: Grid convergence of various quantities.

Table 3 2DB: Observed order of accuracy on the three finest grids.

	SENSEI second-order	SENSEI mixed-order	CFL3D	FUN3D
Observed Order, C_L	3.66	4.65	0.99	1.44
Observed Order, C_D	2.14	2.83	2.37	oscillatory
Observed Order, $C_{D,p}$	2.95	2.89	2.87	2.61
Observed Order, $C_{D,v}$	0.54	oscillatory	0.89	0.64

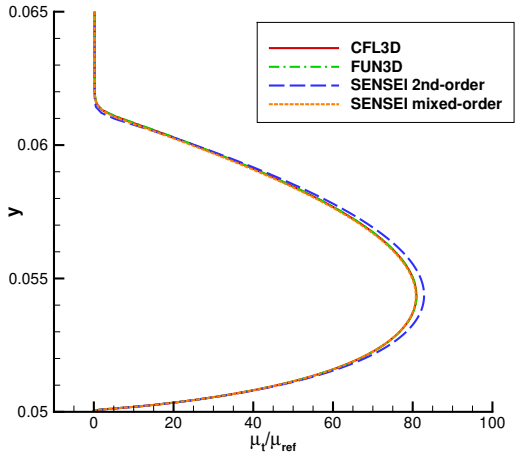


Fig. 12 2DB: Turbulent eddy viscosity through boundary layer at $x = 0.75$.

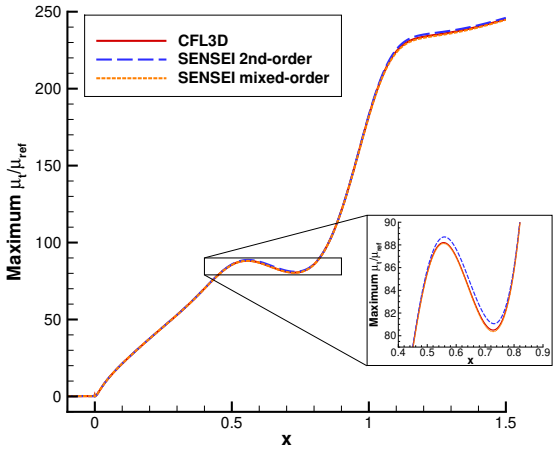
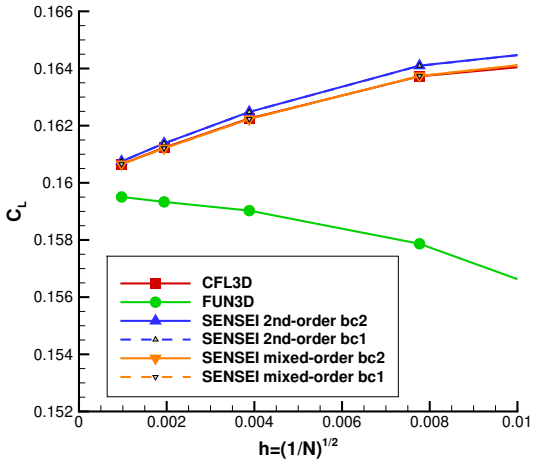


Fig. 13 2DB: Maximum value of turbulent eddy viscosity through the boundary layer along the bump.

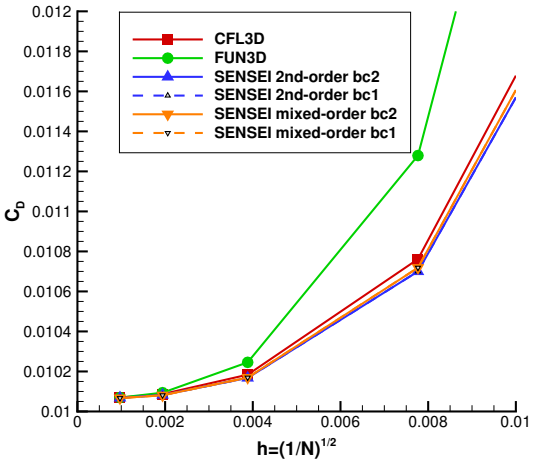
4. 2DANW

The 2D airfoil near wake case (2DANW) is flow over a DSMA661 (model A) airfoil. The grid convergence for the lift, pressure drag, viscous drag, and total drag are shown in Fig 14 and the observed order of accuracy is shown in Table 4. Again, the discretization order of the turbulence equation only has a slight effect on the solution on the finer grids. There is a difference of only 0.06% in the lift on the finest grid level between the second-order and mixed-order solutions. It should also be noted that when the same discretization is used as CFL3D (mixed-order) the lift is very similar. Unlike the 2DB, the boundary condition order has effectively no impact on the quantities of interest for this problem with less than 0.005% difference in lift on all grid levels. This is likely because there are no other boundaries anywhere near the airfoil for this case (ignoring the interblock boundaries at the trailing edge). Because the boundary conditions have very little effect on the solution, the rest of the results for this case will be presented using the second-order boundary condition.

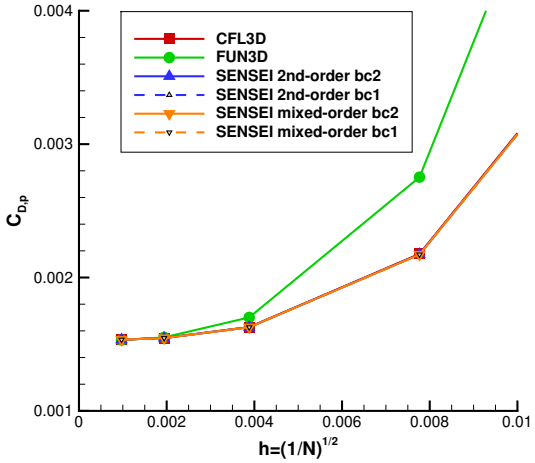
The pressure and and skin friction distributions across the surface of the airfoil are presented in Fig. 15. There is no discernible difference between all of the solutions except in certain regions where the distribution is not smooth. This nonsmoothness in the pressure and skin friction distributions has been attributed to the nonsmooth nature of the airfoil geometry especially near the leading edge. Fig. 15a contains an insert focusing on one of these oscillations to show the differences in skin friction between the three codes. Another unique feature of this problem is the wake. The velocity profile across the wake is measured at several stations downstream of the airfoil and are plotted in Fig. 16. The wake deficit at the various stations match very well, with SENSEI matching the CFL3D results within 1.4%.



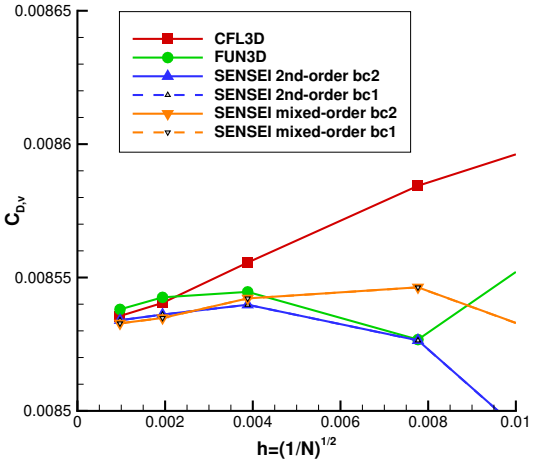
(a) Convergence of C_L.



(b) Convergence of C_D.



(c) Convergence of $C_{D,p}$.

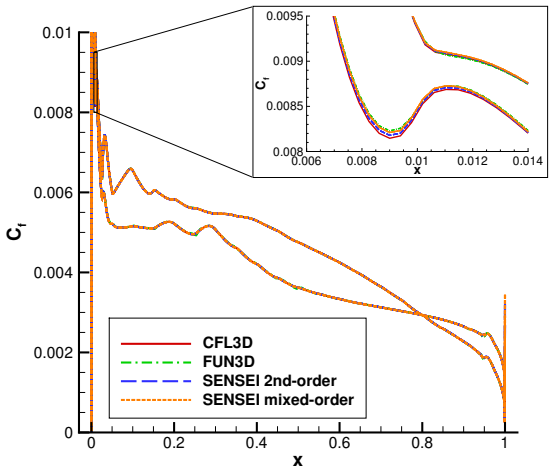


(d) Convergence of $C_{D,v}$.

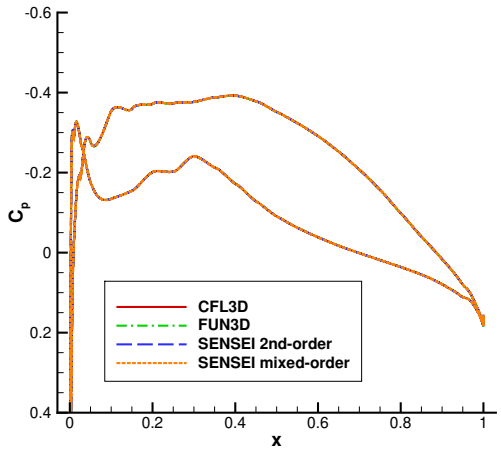
Fig. 14 2DANW: Grid convergence of various quantities.

Table 4 2DANW: Observed order of accuracy on the three finest grids.

	SENSEI second-order	SENSEI mixed-order	CFL3D	FUN3D
Observed Order, C_L	0.81	0.88	0.83	0.83
Observed Order, C_D	2.56	2.64	2.49	2.63
Observed Order, C_D, p	2.72	2.74	2.72	2.91
Observed Order, C_D, v	1.80	1.86	1.62	negative



(a) C_f distribution on finest mesh with inset highlighting differences.



(b) C_p distribution on finest mesh.

Fig. 15 2DANW: Grid convergence of various quantities.

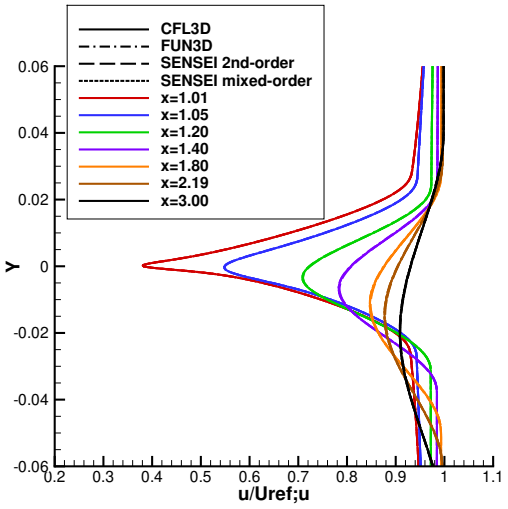


Fig. 16 u-velocity at different stations across the wake.

VII. Conclusions & Future Work

This paper has presented the work done in SENSEI to implement and verify the turbulence models and improve the rest of the code base. As part of this development, we began to adopt many object-oriented principles for the new capabilities as well as refactoring some existing capabilities to fit this paradigm. Implementing code in an object-oriented manner increases the modularity and encapsulation of the code and it can reduce the difficulty and time required to add new capabilities through polymorphism. The new object oriented paradigm was extended to several portions of the code base including the linear system and boundary conditions, which have benefited greatly from this programming style. The governing equations were nondimensionalized resulting in an eight order of magnitude reduction in the LHS condition number. This reduction in the condition number of the LHS allows much more accurate solves of the linear system and allows much higher CFL numbers to be run stably. The hand-coded residual Jacobian was checked using finite differences to ensure that it was implemented correctly.

We also outlined our implementation of the SA-neg turbulence model in SENSEI and the turbulence modeling framework as a whole. Verification was performed using four 2D cases from the NASA Langley Turbulence Modeling Resource website: the zero pressure gradient flat plate, the coflowing jet, the bump-in-channel, and the airfoil near-wake cases. These cases were run in SENSEI, and the results were compared with results from CFL3D and FUN3D. All of the results seemed to be converging to the same answers as the published results for FUN3D and CFL3D, which have been rigorously verified. Matching the other codes, gives some amount of confidence that SENSEI's implementation of the SA-neg model is correct.

However, while presenting the results for these cases, some differences were highlighted. The most significant of these differences were attributed to using different-order spatial discretizations of the turbulence equations and using different-order boundary conditions. The TMR results for FUN3D and CFL3D both use a first-order discretization of the SA equation while SENSEI has the ability to run both the mean flow and turbulence equations with either mixed- or second-order discretizations. It was noted that for all of the cases, running fully second-order as opposed to mixed-order did have a slight effect on the eddy viscosity. When a second-order spatial discretization of the turbulence equation was used, the edge of the turbulent region was more sharp compared to the mixed-order solution. This was attributed to less numerical dissipation smearing out the turbulence working variable at the edge of the turbulent region. It was also noted that the maximum value of the second-order eddy viscosity was different than it was in the mixed-order solutions for some cases such as the coflowing jet and airfoil. However, these small differences in eddy viscosity did not have much of an effect on the mean flow properties or the viscous drag except on the coarsest meshes.

The order of the boundary conditions also had a significant effect on the final solution for most of the problems. When the inflow or outflow boundaries were close to or intersected a wall, like the flat plate and coflowing jet cases, the boundary condition order affected the local skin friction in a significant manner. This is more pronounced when the boundary is at the leading edge of the wall like in the coflowing jet problem, where the low-order boundary condition affects the skin friction along the entire plate and thus the integrated drag coefficient. When the wall intersects an outflow boundary, like in the flat plate case, the effect of the first-order boundary condition is more localized and thus does not significantly affect the integrated drag. Thus, for problems where walls of interest intersect inflow or outflow boundaries, it is more important to use higher-order boundary conditions. When the other boundaries were farther away the body of interest, like the bump case, the order of the boundary conditions had only a slight impact on the skin friction, but a larger impact on the pressure distribution. Finally, when the boundaries were very far (500 body lengths) away from the body, as in the airfoil case, the order of the boundary conditions had negligible effect. If SENSEI is run using a mixed-order spatial discretization, similar to CFL3D, then the results between CFL3D and SENSEI match extremely well.

In the future, we hope to extend this work to other turbulence models and higher dimensional problems. Because of the turbulence framework that has been implemented, future developers can easily add any eddy viscosity turbulence model. Future studies should continue to explore the differences between the solutions using different boundary conditions. These boundary condition differences include not only the order of the boundary conditions but also the specific implementations (strong versus weak, what variables are extrapolated to the outer state, etc). Another aspect of the boundary conditions that could be considered is performing the boundary extrapolations in physical space rather than computational space. It would also be interesting to continue examining the effect of the spatial discretization order on the final solution in more detail and with more complex problems, for SA as well as other turbulence models. For these studies, it would be important to look at the effect of the discretization order on the turbulence variables as well as the effect on the mean flow and on engineering quantities of interest such as lift and drag.

Appendix A

A. Favre-Averaged Navier-Stokes Equations

The Favre-averaged Navier-Stokes equations (FANS) describe the motion of a compressible, turbulent fluid and are a statement of conservation of mass, momentum, and energy [7]. To derive the FANS, the variables in the Navier-Stokes equations (NS) are decomposed into mean and fluctuating parts. The mean is calculated using a density-weighted average in time, as opposed to Reynolds averaging, which does not use density-weighting. After performing the Favre

averaging, the FANS equations can be written as

$$\begin{aligned}
\frac{\partial \bar{\rho}}{\partial t} + \frac{\partial}{\partial x_i}\left(\bar{\rho}\tilde{u}_i\right) &= 0, \\
\frac{\partial}{\partial t}\left(\bar{\rho}\tilde{u}_i\right) + \frac{\partial}{\partial x_j}\left(\bar{\rho}\tilde{u}_j\tilde{u}_i\right) &= -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\bar{t}_{ji} - \overline{\rho u_j'' u_i''}\right], \\
\frac{\partial}{\partial t}\left[\bar{\rho}\left(\tilde{e} + \frac{\tilde{u}_i\tilde{u}_i}{2}\right) + \frac{\overline{\rho u_i'' u_i''}}{2}\right] + & \\
\frac{\partial}{\partial x_j}\left[\bar{\rho}\tilde{u}_j\left(\tilde{h} + \frac{\tilde{u}_i\tilde{u}_i}{2}\right) + \tilde{u}_j\frac{\overline{\rho u_i'' u_i''}}{2}\right] &= \frac{\partial}{\partial x_j}\left[-q_{L_j} - \overline{\rho u_j'' h''} + \overline{t_{ji}u_i''} - \overline{\rho u_j''\frac{1}{2}u_i'' u_i''}\right] \\
&\quad + \frac{\partial}{\partial x_j}\left[\tilde{u}_i\left(\bar{t}_{ij} - \overline{\rho u_i'' u_j''}\right)\right].
\end{aligned} \tag{22}$$

The extra terms that appear compared to the NS equations are correlations between instantaneous fluctuating quantities. If these correlations are known, the FANS equations can be solved exactly. Most often, these correlation terms are not known and must instead be modeled. Once these terms are modeled, the system is closed with an equation of state. In this work, the system is closed using the perfect gas assumption

$$\tilde{p} = \bar{\rho}R\tilde{T}. \tag{23}$$

To simplify the notation in Eq. 22, the following definitions are made. First, the Reynolds stress tensor, τ_{ij}, may be defined as

$$\bar{\rho}\tau_{ij} \equiv -\overline{\rho u_i'' u_j''}. \tag{24}$$

Next, the kinetic energy per unit volume is defined as

$$\bar{\rho}k \equiv \frac{1}{2}\overline{\rho u_i'' u_i''}. \tag{25}$$

Finally, the turbulent transport of heat may be defined as

$$q_{T_j} \equiv \overline{\rho u_j'' h''}. \tag{26}$$

With the definitions given in Eqs. 24 – 26, the FANS equations can be rewritten as

$$\begin{aligned}
\frac{\partial \bar{\rho}}{\partial t} + \frac{\partial}{\partial x_i}\left(\bar{\rho}\tilde{u}_i\right) &= 0, \\
\frac{\partial}{\partial t}\left(\bar{\rho}\tilde{u}_i\right) + \frac{\partial}{\partial x_j}\left(\bar{\rho}\tilde{u}_j\tilde{u}_i\right) &= -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\bar{t}_{ji} + \overline{\rho}\tau_{ji}\right], \\
\frac{\partial}{\partial t}\left(\bar{\rho}e_t\right) + \frac{\partial}{\partial x_j}\left(\bar{\rho}\tilde{u}_j h_t\right) &= \frac{\partial}{\partial x_j}\left[-q_{L_j} - q_{T_j} + \overline{t_{ji}u_i''} - \overline{\rho u_j''\frac{1}{2}u_i'' u_i''}\right] \\
&\quad + \frac{\partial}{\partial x_j}\left[\tilde{u}_i\left(\bar{t}_{ij} + \overline{\rho}\tau_{ij}\right)\right],
\end{aligned} \tag{27}$$

where the total energy, e_t, and total enthalpy, h_t, are given as

$$e_t = \tilde{e} + \frac{1}{2}\tilde{u}_i\tilde{u}_i + k, \quad h_t = \tilde{h} + \frac{1}{2}\tilde{u}_i\tilde{u}_i + k, \tag{28}$$

and k is the turbulent kinetic energy.

B. Compressible Flow Closure Approximations

In the present form, Eq. 27 cannot be solved; all of the correlation terms must be modeled in some manner. The following compressible flow closure approximations are made to model the Reynolds stress tensor, turbulent heat flux, and the molecular diffusion and transport of turbulence [7].

1. Reynolds-Stress Tensor

The Boussinesq approximation is used to model the Reynolds stress tensor. The Boussinesq approximation states that the Reynolds stress is directly proportional to the strain rate of the mean flow through the turbulent eddy viscosity, μ_T. The Reynolds stress tensor is given by

$$\bar{\rho}\tau_{ij} = 2\mu_T \left(S_{ij} - \frac{1}{3}\frac{\partial \tilde{u}_k}{\partial x_k}\delta_{ij} \right) - \frac{2}{3}\bar{\rho}k\delta_{ij}, \tag{29}$$

where S_{ij} is the mean strain rate,

$$S_{ij} = \frac{1}{2}\left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i} \right). \tag{30}$$

2. Turbulent Heat-Flux

Turbulent heat flux is modeled using a Reynolds analogy where the turbulent heat flux is assumed to be proportional to the mean temperature gradients

$$q_{T_j} = -\frac{\mu_T C_p}{Pr_T}\frac{\partial \tilde{T}}{\partial x_j}, \tag{31}$$

where Pr_T is the turbulent Prandtl number, C_p is the specific heat at constant pressure, and μ_T is the turbulent eddy viscosity.

3. Molecular Diffusion and Turbulent Transport

The molecular diffusion and turbulent transport terms are often lumped together and are modeled as

$$\overline{t_{ji}u_i''} - \overline{\rho u_j'' \frac{1}{2} u_i'' u_i''} = \left(\mu + \frac{\mu_T}{\sigma_k} \right) \frac{\partial k}{\partial x_j}, \tag{32}$$

where σ_k is a constant, which depends upon how the turbulent kinetic energy is modeled. In some turbulence models, this lumped term may take a slightly different form. In other turbulence models, such as the Spalart-Allmaras model, the molecular diffusion and turbulent transport may not even be modeled; in such cases, this term may be neglected.

Appendix B

Listing 1 Outline of `turb_model_t`

```
type, abstract :: turb_model_t

  ! Number of turbulence variables
  integer :: n_turb

  ! Number of miscellaneous variables
  integer :: n_misc

contains

  public

  ! Allocates as sets the turb_init aray
  procedure(set_init_i),        nopass, deferred :: set_turb_init

  ! Turbulent kinetic energy (TKE) for the model
  procedure(tke_i),             nopass, deferred :: calc_tke
  procedure(scalar_jac_i),      nopass, deferred :: calc_tke_jacobian
  procedure(scalar_mean_jac_i), nopass, deferred :: calc_tke_mean_jacobian

  ! Turbulent eddy viscosity (mut) for the model
  procedure(mut_i),             nopass, deferred :: calc_mut
  procedure(scalar_jac_i),      nopass, deferred :: calc_mut_jacobian
  procedure(scalar_mean_jac_i), nopass, deferred :: calc_mut_mean_jacobian

  ! Molecular diffusion / turbulent transport (MDTT) for the model
  procedure(mdtt_i),            nopass, deferred :: calc_mdtt
```

```
  procedure(matrix_jac_i),      nopass, deferred :: calc_mdtt_jacobian
  procedure(matrix_mean_jac_i), nopass, deferred :: calc_mdtt_mean_jacobian

  ! Effective viscosity (mueff) for the turbulent viscous flux
  procedure(mueff_i),           nopass, deferred :: calc_mueff
  procedure(vector_jac_i),      nopass, deferred :: calc_mueff_jacobian

  ! Production term for the model
  procedure(src_terms_i),       nopass, deferred :: calc_production
  procedure(vector_jac_i),      nopass, deferred :: calc_production_jacobian

  ! Destruction term for the model
  procedure(src_terms_i),       nopass, deferred :: calc_destruction
  procedure(vector_jac_i),      nopass, deferred :: calc_destruction_jacobian

  ! Diffusion term for the model
  procedure(src_terms_i),       nopass, deferred :: calc_diffusion
  procedure(stencil_jac_i),     nopass, deferred :: calc_diffusion_jacobian

  ! Source terms for the model
  procedure(turb_source_i),     nopass, deferred :: calc_source
  procedure(stencil_jac_i),     nopass, deferred :: calc_source_jacobian

  ! Computes gradients of the turbulence variables
  procedure,                                     :: calc_turb_grad

  ! Various terms of interest for the model
  procedure(misc_terms_i),      nopass, deferred :: turb_misc
  procedure(misc_names_i),      nopass, deferred :: turb_misc_name

  ! Tecplot output
  procedure,                                     :: setup_tecplot_turb
  procedure,                                     :: write_tecplot_turb

  ! Reynolds stress at a given cell
  procedure                                      :: reynolds_stress

  ! Converts turbulence variables to and from dimensional form
  procedure(dimen_i),           nopass, deferred :: dimen_to_nondimen
  procedure(dimen_i),           nopass, deferred :: nondimen_to_dimen

  ! Deallocates the turbulence model
  procedure                                      :: destroy

end type turb_model_t
```

References

[1] Derlaga, J. M., Phillips, T., and Roy, C. J., "SENSEI Computational Fluid Dynamics Code: A Case Study in Modern Fortran Software Development," AIAA Paper 2013-2450, 21st AIAA Computational Fluid Dynamics Conference, San Diego, California, Jun. 2013. doi:10.2514/6.2013-2450.

[2] Spalart, P. R., and Allmaras, S. R., "A one-equation turbulence model for aerodynamic flows," AIAA Paper 1992-439, 30th Aerospace Sciences Meeting and Exhibit, Reno, NV, Jan. 1992. doi:10.2514/6.1992-439.

[3] Allmaras, S. R., Johnson, F. T., and Spalart, P. R., "Modifications and Clarifications fo rthe Implementation of the Spalart-Allmaras Turbulence Model," ICCFD 7-1902, Big Island, Hawaii, Jul. 2012. URL `http://iccfd.org/iccfd7/assets/pdf/papers/ICCFD7-1902_paper.pdf`.

[4] Rumsey, C. L., Smith, B. R., and Huang, G. P., "Description of a Website Resource for Turbulence Modeling Verification and Validation," AIAA Paper 2010-4742, 40th Fluid Dynamics Conference and Exhibit, Chicago, Illinois, June 2010. doi:10.2514/6.2010-4742.

[5] Krist, S. L., Biedron, R. T., and Rumsey, C. L., "CFL3D Users Manual (Version 5.0)," NASA/TM 2008-2155370, Nov. 1998.

[6] Biedron, R. T., Carlson, J.-R., Derlaga, J. M., Gnoffo, P. A., Hammond, D. P., Jones, W. T., Kleb, B., Lee-Rausch, E. M., Nielsen, E. J., Park, M. A., Rumsey, C. L., Thomas, J. L., and Wood, W. A., "FUN3D User's Manual Version 13.3," NASA/TM 2018-219808, NASA, 2018.

[7] Wilcox, D., *Turbulence Modeling for CFD*, 3rd ed., DCW Industries, La Canada, CA, 2010.

[8] Rumsey, C. L., "Apparent transition behavior of widely-used turbulence models," *International Journal of Heat and Fluid Flow*, Vol. 28, No. 6, 2007, pp. 1460–1471. doi:10.1016/j.ijheatfluidflow.2007.04.003.

[9] Menter, F. R., "Two-equation eddy-viscosity turbulence models for engineering applications," *AIAA Journal*, Vol. 32, No. 8, 1994, pp. 1598–1605. doi:10.2514/3.12149.

[10] van Leer, B., "Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method," *Journal of Computational Physics*, Vol. 32, No. 1, 1979, pp. 101–136. doi:10.1016/0021-9991(79)90145-1.

[11] Roe, P., "Approximate Riemann solvers, parameter vectors, and difference schemes," *Journal of Computational Physics*, Vol. 43, No. 2, 1981, pp. 357–372. doi:10.1016/0021-9991(81)90128-5.

[12] Jameson, A., Schmidt, W., and Turkel, E., "Numerical solution of the Euler equations by finite volume methods using Runge Kutta time stepping schemes," AIAA Paper 1981-1259, 14th Fluid and Plasma Dynamics Conference, Palo Alto, CA, Jun. 1981. doi:10.2514/6.1981-1259.

[13] Jalali, A., and Ollivier Gooch, C. F., "Higher-order Unstructured Finite Volume Methods for Turbulent Aerodynamic Flows," AIAA Paper 2015-2284, 22nd AIAA Computational Fluid Dynamics Conference, 2015. doi:10.2514/6.2015-2284.

[14] Burgess, N., and Mavriplis, D., "High-order Discontinuous Galerkin Methods for Turbulent High-lift Flows," ICCFD 7-4202, Big Island, Hawaii, Jul. 2012. URL `http://www.iccfd.org/iccfd7/assets/pdf/papers/ICCFD7-4202_paper.pdf`.

[15] Toro, E. F., *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction*, 3rd ed., Springer Science + Business Media, New York, NY, 2009. doi:10.1007/b79761.

[16] Saad, Y., and Schultz, M. H., "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM Journal on Scientific and Statistical Computing*, Vol. 7, No. 3, 1986, pp. 856–869. doi:10.1137/0907058.

[17] van der Vorst, H. A., "Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems," *SIAM Journal on Scientific and Statistical Computing*, Vol. 13, No. 2, 1992, pp. 631–644. doi:10.1137/0913035.

[18] Burgess, N. K., "An adaptive discontinuous Galerkin solver for aerodynamic flows," Ph.D. Thesis, University of Wyoming, 2011.

[19] Tyson, W. C., "On Numerical Error Estimation for the Finite-Volume Method with an Application to CFD," Ph.D. Thesis, Virginia Polytechnic Institute and State University, Oct. 2018.

[20] Rumsey, C. L., and Thomas, J. L., "Application of FUN3D and CFL3D to the Third Workshop on CFD Uncertainty Analysis," NASA/TM 1998-208444, June 2008.