Oral presentation | Incompressible/compressible/hypersonic flow
Incompressible/compressible/hypersonic flow-II
Wed. Jul 17, 2024 4:30 PM - 6:30 PM Room D

[9-D-02] Solving High Reynolds Number Flows on Cartesian Cut-cell Meshes using a Jacobian-Free Newton-Krylov Method

*Alexander O. Kleb[1], Krzysztof J. Fidkowksi[1], Joaquim R. R. A. Martins[1] (1. University of Michigan)

Keywords: RANS, wall-modeled, cut-cell

Twelfth International Conference on
Computational Fluid Dynamics (ICCFD12),
Kobe, Japan, July 14-19, 2024

Solving High Reynolds Number Flows on Cartesian Cut-cell Meshes using a Jacobian-Free Newton–Krylov Method.

Alex Kleb*, Krzysztof J. Fidkowski* and Joaquim R. R. A. Martins*
Corresponding author: akleb@umich.edu
* University of Michigan, USA.

Abstract: In this work, we developed a Newton–Krylov method for a second-order Cartesian cut cell Reynolds-averaged Navier–Stokes (RANS) solver, Viscous Aerodynamic Cartesian Cut cells (VACC), with the one equation Spalart–Allmaras (SA) turbulence model. The Newton–Krylov method uses pseudo-transient continuation and a point Jacobi preconditioner to accelerate convergence. Then various wall functions were compared on a finite flat plate and 2D bump cases. The SA analytical wall function was used as a baseline. An ordinary differential equation (ODE) wall function and wall-modeled RANS (WMRANS) approach were also implemented. Although these methods all showed promise, the interior viscous fluxes resulted in oscillatory pressures. These oscillations degraded the accuracy of all of the solutions. Additional work needs to be done to strength the viscous stencil in and around the cut cells.

Keywords: Cartesian Cut Cells, Computational Fluid Dynamics, Turbulence Modeling.

1 Introduction

The Reynolds-averaged Navier–Stokes (RANS) equations are commonplace in aerodynamic analysis and design. They provide acceptable accuracy for a reasonable cost at aircraft cruise conditions. The development of the RANS adjoint encouraged the extensive use of RANS as the analysis tool in aerodynamic shape optimization (ASO) [1, 2, 3, 4].

One of the most important and time-consuming parts of a RANS solution workflow is the generation of the computational mesh. The computational mesh needs to be carefully crafted to balance the cost and accuracy of a simulation. When performing aerodynamic shape optimization (ASO), additional care is required to ensure that the mesh can be warped by a numerical optimizer [5]. With these three criteria in mind, generating a suitable mesh by hand can take weeks or even months. While RANS solution methods have become more efficient, the mesh generation process for complex geometries remains the most time-intensive part of the solution procedure [6].

Cartesian cut-cell methods circumvent this problem by automatically generating meshes without user intervention [7]. Accurate computational meshes for complex geometries can be generated in a matter of seconds or minutes [8]. A wetted surface geometry is cut out of a background Cartesian mesh, and then adaptive mesh refinement (AMR) is used to reduce discretization errors to adequate levels. This procedure provides an accurate solution with minimal user input or a priori assumptions about the flow structure.

Unfortunately, automatic meshing methods are currently only feasible for inviscid or incompressible flows [9, 10, 11, 12]. When performing high-fidelity simulations, viscous effects such as skin friction can contribute significantly to drag calculations and aircraft design. The main challenge for the Cartesian cut-cell method is resolving boundary layers [12]. The flow states change much faster in the off-wall direction than the wall-aligned direction. Traditional, boundary conforming, meshes address this issue by stretching cells or grid points in the wall-aligned direction. This anisotropy allows the mesh to resolve the rapidly changing off-wall properties without wasting too many grid points in the wall-aligned direction. Isotropic cells small enough to resolve the off-wall properties would introduce too many superfluous points along the wall, making these meshes computationally intractable. Alternatively, cut-cells could be stretched along the boundary, but this approach can compromise the automatic nature of the meshing algorithm for complex geometries.

Wall functions may provide a method for accurate solutions of Cartesian cut-cell meshes without sacrificing automatic meshing. Wall functions provide a sub-mesh scale approximation for the flow in the boundary layer [13]. This allows significantly larger cells than in boundary conforming, wall-resolved

meshes. Typical wall functions present an analytical function that can be tuned to match the flow at a 'forcing point' some distance away from the wall. The wall function can be used to get wall tangential velocities and gradients to augment flux computations.

The accuracy of wall functions can break down depending on the location of the 'forcing point' in the boundary layer. A quadratic function can be fit to the boundary layer using two flow states and the no-slip condition at the wall [12, 14]. However, this 'wall function' would only be applicable in the viscous sublayer of the boundary layer, and it therefore does not provide much benefit over a wall-resolved mesh. Allmaras developed an analytical wall function that matches the solution of the RANS equations with a Spalart–Allmaras (RANS-SA) turbulence model in [12]. This provides the exact, wall-resolved, solution for a RANS-SA boundary layer with no pressure gradient through the log region. The convection terms in the wake region cause this model to break down past the log region.

An important part of using a wall function is ensuring that the forcing point lies within a valid region of the wall function. The forcing point location can introduce issues for automatic mesh generation on geometries as simple as an airfoil. The thickness of the boundary layer on the suction side can be orders of magnitude larger than the thickness on the pressure side. Being able to consistently place the forcing point in the proper region of the boundary layer without over-resolving the background mesh is challenging. Additionally, some wall functions, such as Spalding's wall function may lose accuracy in the transition between the viscous sublayer and the log region. It is therefore possible that a finer mesh is less accurate than a coarser one when using a wall function.

Berger and Aftosmis [15] developed a unique type of wall function that solves a one-dimensional ordinary differential equation (ODE) instead fitting data to a predetermined functional. Instead of being fit to one or two parameters, the forcing point data are used as a boundary condition in the ODE. This model introduces an approximation for the convection term in the wake region, allowing the model to remain valid even further from the wall than the SA wall function.

Ursachi et al. [16] developed a new wall model RANS (WMRANS) approach that circumvents the need for wall functions entirely. It enforces the same total shear stress, but simplifies the velocity and turbulence variable profiles. Near the wall, the turbulent viscosity approaches a constant rather than being linear in the log layer and quartic in the viscous sublayer. This generates a velocity profile that is linear near the wall. Since the velocity no longer under goes rapid nonlinear changes near the wall, coarser meshes are acceptable. Computational volumes with linear representations of the solution do a much better job matching the analytical solution to the wall function. This method has the additional benefit that it does not require forcing points, making it much easier to implement.

In this paper, we compare the effectiveness of the SA wall function, the ODE wall function and the equivalent shear-stress boundary condition methods. Section 2 presents an in depth documentation of the governing equations involved in each method. Section 3 outlines our Jacobian-free Newton–Krylov (JFNK) RANS-SA solver and the way each boundary method fits in. Three exterior flow cases are compared in section 4. Finally, concluding remarks are left to section 5.

2 Governing Equations

In this work, we investigate the Reynolds-averaged Navier–Stokes (RANS) equations with the compressible Spalart–Allmaras negative (SA-neg) turbulence model,

$$\partial_t(\rho\tilde{\nu}) + \partial_j(\rho u_j\tilde{\nu}) = \partial_j\left(\frac{1}{\sigma}\rho(\nu+\nu')\partial_j\tilde{\nu}\right) - \frac{1}{\sigma}(\nu+\nu')\partial_j\rho\partial_j\tilde{\nu} + \frac{c_{b2}\rho}{\sigma}\partial_j\tilde{\nu}\partial_j\tilde{\nu} + P - D. \tag{1}$$

We leave the detailed explanation of each term to numerous online resources such as the NASA turbulence modeling resource[1].

To accurately resolve boundary layers, we make use of three different wall models. The first two wall models investigated follow the work of Berger and Aftosmis [12, 15]. The first wall function is a traditional analytical function that is fit based on the background state at an interior forcing point. This function was developed by Allmaras [12] to match a background, wall-resolved, RANS-SA solution into the log region of the boundary layer. The analytical wall function is given by,

$$\begin{aligned} u^+(y^+) = \bar{B} + c_1\log\left(\left(y^+ + a_1\right)^2 + b_1^2\right) - c_2\log\left(\left(y^+ + a_2\right)^2 + b_2^2\right) \\ - c_3\,\mathrm{atan2}\left(b_1, y^+ + a_1\right) - c_4\,\mathrm{atan2}\left(b_2, y^+ + a_2\right), \end{aligned} \tag{2}$$

[1] https://turbmodels.larc.nasa.gov/

where the given coefficients given are from Allmaras [12] and the plus quantities are given by,

$$u^+ = \frac{u}{u_\tau} \quad \text{and} \quad y^+ = \frac{\eta u_\tau}{\nu}, \tag{3}$$

where u_τ is the friction velocity, u is the wall tangential velocity, and η is the wall normal distance. The second wall function retains the forcing point, but solves a 1D boundary valued ODE instead of fitting a state to the function. This wall function was developed by Berger and Aftosmis [15] to increase accuracy into the wake region of the boundary layer. The ODE is given by,

$$\begin{aligned} \frac{\partial}{\partial \eta}\left[(\mu+\mu_t)\frac{\partial u}{\partial \eta}\right] &= \left.\frac{\partial p}{\partial \xi}\right|_F + \psi(\eta)\ \rho|_F \left[u_F \left.\frac{\partial u}{\partial \xi}\right|_F + v_F \left.\frac{\partial u}{\partial \eta}\right|_F\right] \\ \frac{\partial}{\partial \eta}\left[\frac{1}{\sigma}(\nu+\tilde{\nu})\frac{\partial \tilde{\nu}}{\partial \eta}\right] &= -\frac{c_{b2}}{\sigma}\left(\frac{\partial \tilde{\nu}}{\partial \eta}\right)^2 - (P-D), \end{aligned} \tag{4}$$

where ξ is the wall tangential distance, u is the tangential velocity, v is the wall normal velocity, μ is the dynamic viscosity, μ_t is the turbulent dynamic viscosity, ν is the kinematic viscosity, $\tilde{\nu}$ is the turbulence variable, p is the pressure, $(\cdot)|_F$ is a constant quantity evaluated at the forcing point, and $(P-D)$ along with σ and c_{b2} come from the definitions in the SA turbulence model. $\psi(\eta)$ is an activation function that turns on the convection terms in the ODE away from the wall. We define our activation function as presented by Berger and Aftosmis [15],

$$\psi(\eta) = \frac{u^+_{\mathrm{SA}}(\eta)}{u^+_{\mathrm{SA}}(F)}, \tag{5}$$

where u^+_{SA} is defined by equation 2.

The final wall model modifies the SA turbulence model to generate a linear velocity profile near the wall instead of the traditional nonlinear velocity profile. It does not need to introduce data at an interior forcing point to do this. This model was developed by Ursachi et al. [16]. The changes to the SA model are as follows,

$$\mu_t = \rho \nu f_{m1}, \tag{6a}$$

$$f_{m1} = \sqrt{\chi^2 + \chi^2_{t0}}, \tag{6b}$$

$$\tilde{S} = S + \frac{\tilde{\nu}}{\kappa^2 d^2} f_{m2}, \tag{6c}$$

$$f_{m2} = 1 - \frac{\chi}{1+f_{m1}}, \tag{6d}$$

where χ_{t0} is a parameter that indicates how fast the wall-modeled solution approaches the RANS solution in the log layer. f_{m1} has replaced f_{v1} and f_{m2} has replaced f_{v2}. The resulting velocity profile does not go to zero at the wall and necessitates updated boundary conditions. For the momentum equations we have a Robin condition between the wall shear stress and slip velocity,

$$\vec{\tau}_{w,m} = \rho\, |\vec{u}_{\tau,m}|\, \vec{u}_{\tau,m}, \tag{7}$$

where $u_{\tau,m}$ is the friction velocity,

$$\vec{u}_{\tau,m} = \frac{\vec{u}_\parallel}{u^+_{\mathrm{WM}}(0)}, \tag{8}$$

where $\vec{u}_\parallel$ is the wall tangential velocity and u^+_{WM} is analytical wall model velocity. We leave the full definition of u^+_{WM} to reference [16], but reproduce the relevant result here,

$$u^+_{\mathrm{WM}}(0) = \frac{\log(\kappa \chi_{t0})}{\kappa} + c \tag{9}$$

where κ is the von Kŕmń constant, and c is defined in [16]. The wall model also affects the energy equation. Ursachi et al. derived the heat flux of the wall model given the RANS-SA heat flux,

$$q_w|_{\mathrm{WM}} = q_w|_{\mathrm{RANS}} - \vec{u}_\parallel \cdot \vec{\tau}_{w,m}. \tag{10}$$

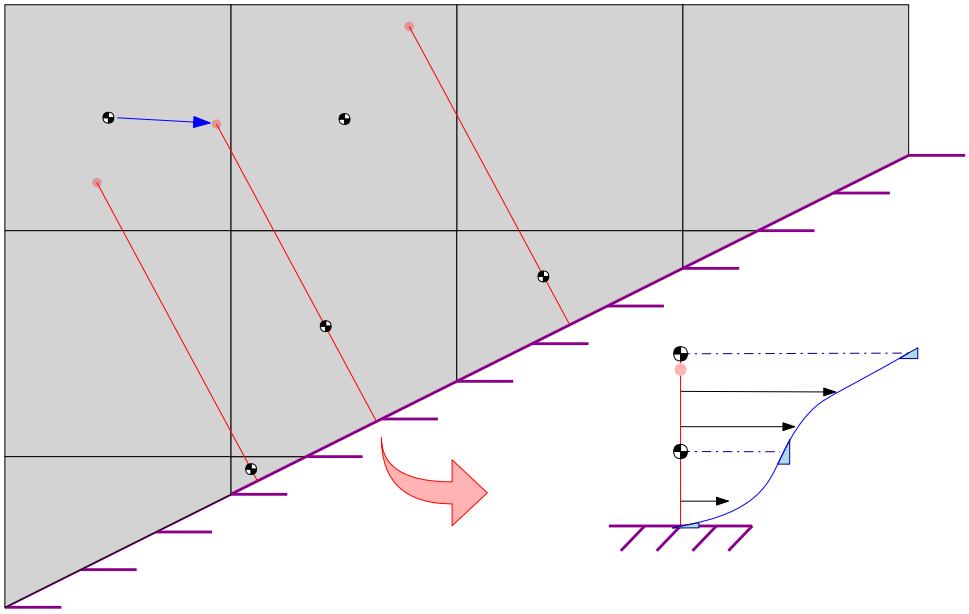


Figure 1: The coupling between the forcing point and the background mesh. The background mesh provides a state at the forcing point that is fit to the desired wall function. The wall function can then provide states and gradients at various locations for computing viscous fluxes.

Since there is a slip velocity, the heat flux for an adiabatic boundary condition is not zero. This is not a physical condition, as the nonzero heat flux accounts for the heat released by viscous dissipation in the unresolved near-wall region.

3 Methodology

Our flow solver is a two-dimensional second-order finite volume Cartesian cut-cell method [14, 17]. We retain second-order inviscid fluxes for the turbulence variable. Cell gradients are computed using a least-squares reconstruction from neighboring cell states. Limiters are computed by solving a local linear programming problem in each cell to obtain a directional limiter that retains as much of the original gradient as possible while satisfying total variation diminishing constraints.

The introduction of viscous terms necessitates gradients at faces. For faces between two interior cells, the cell centroids are aligned with their neighbors along either the x or y axis through the face centroid. Gradients in the direction perpendicular to the face normal are computed by averaging the adjacent cell gradients. The gradients in the direction parallel to the face normal are computed with a center difference across the adjacent cell states.

In cut-cells, the linear state representation is not accurate enough in the isotropic Cartesian cut-cells. To circumvent this issue without over-resolving the wall tangential features, we use one of the wall models in section 2. WMRANS presented by Ursachi et al. solves the issue by making the velocity profile near the wall linear. With a linear velocity profile near the wall, the need for additional resolution is mitigated and the system can be solved similar to inviscid flow problems.

The two wall function methods make use of a coupled forcing point strategy to resolve the wall normal velocity state change. Both wall functions follow the same general strategy shown in figure 1. Each boundary face computes a forcing point by drawing a line that starts at the boundary, passing through the cell centroid a distance $h = 1.5 \times \min(dx, dy)$ away from the boundary. The distance h is a constant to compute smooth skin frictions even with varying cut-cell sizes [13]. During a flux evaluation, a state is projected to the forcing point from the cell containing it using its least-squares gradient. The state at the forcing point is then rotated into wall normal and tangential coordinates and fit to the appropriate wall function. Once the wall function is defined, it can be used to inform the fluxes for the background cut-cell. The wall shear is computed straight from the wall function and then rotated back into the global coordinate frame. As before, the interior faces adjacent to refinement boundaries or cut-cells are computed with averaging in the direction perpendicular to the face normal and a difference in the direction parallel to the face. However, in these cases, the centroids are not

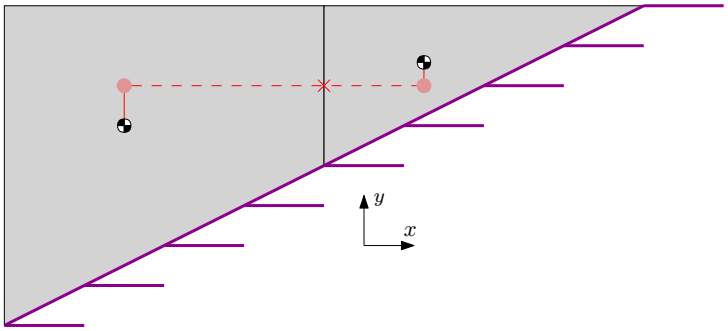


Figure 2: The recentering process used to compute gradients at face centroids not aligned with the cell centroids.

aligned in the coordinate directions through the face centroid. This does not change the gradient that is averaged, but the differenced gradient needs a recentered state so that the difference is computed through the face centroid. Figure 2 shows this process. For the cut cells, this projection is done by reading off states from the associated wall function that is the same distance away from the wall as the projection point. This ignores the state variation in the wall transverse direction, which is generally much smaller. A similar procedure is used to compute the gradient used in averaging, pulling it off the wall function. In the interior cells, this projection is done using the least-squares gradient.

3.1 Jacobian-Free Newton–Krylov Solver

To converge the global solution to steady state, we use a Jacobian-free Newton–Krylov solver with pseudo-transient continuation. A Newton–Krylov method converges a system of residuals to zero. We start with the flux form of the governing PDE,

$$\frac{\partial \mathbf{u}}{\partial t} + \vec{\mathbf{F}}(\mathbf{u}) = \mathbf{S}(\mathbf{u}), \tag{11}$$

where $\mathbf{u}$ is the state vector, $\mathbf{S}$ is the source term and $\vec{\mathbf{F}}$ is the flux vector. For the 2D finite volume method, the steady-state residual for a single cell takes the form,

$$R_{\text{cell}} = \sum_{e=1}^{N_e} \left(\hat{F}\Big|_e \cdot \vec{n}_e \right) l_e - S_{\text{cell}} A_{\text{cell}}, \tag{12}$$

where e indicates edges of the cell and $\hat{F}$ is a numerical flux computed at an edge. The Newton method solves linear systems at each step,

$$\frac{\partial \mathbf{R}}{\partial \mathbf{u}}\bigg|_{\mathbf{u}^n} \Delta \mathbf{u} = -\mathbf{R}(\mathbf{u}^n), \tag{13}$$

where $\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta \mathbf{u}$. Newton's method exhibits quadratic nonlinear convergence once the state is close to the final solution. However, getting close to the final solution for complex CFD cases can be challenging. To address this, we introduce a pseudo-transient continuation term to guide the Newton–Krylov solver towards the solution using unsteady physics.

Pseudo-transient continuation uses an artificial times step as a globalization strategy for the Newton–Krylov method [18, 19, 20]. We use a backward-Euler time step,

$$\frac{\mathbf{A}}{\Delta \mathbf{t}} \left(\mathbf{u}^{n+1} - \mathbf{u}^n \right) + \mathbf{R}\left(\mathbf{u}^{n+1} \right) = \mathbf{0}, \tag{14}$$

where $\mathbf{A}/\Delta \mathbf{t}$ is the diagonal matrix containing the area of the appropriate cell divided by its local time step. We compute the time steps as described in [17]. While this time step is computed for conserved variables, the Newton solver does not require that we use conserved variables. Our flow solver uses primitive state variables, $[\rho, u, v, p, \tilde{\nu}]$, to obtain less diffusive gradients during the inviscid flux calculation. The computed residuals are still in 'conserved form', but the actual evolution of the state does not matter as long as the scheme is stable and the solver reaches $\mathbf{R}(\mathbf{u}) = \mathbf{0}$. Backward Euler is A-plane stable and therefore will always be stable as long as we have negative real-part eigenvalues.

We apply a nonlinear preconditioner to the artificial time step to match the primitive state update $\Delta\mathbf{u}$ to the conserved state time step, $\mathbf{A}/\Delta\mathbf{t}$. This is not strictly necessary, but we found that it helped nonlinear convergence. The nonlinear preconditioner is,

$$P_{t,\text{local}} = \frac{\partial \mathbf{u}_{\text{conservative}}}{\partial \mathbf{u}_{\text{primitive}}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ u & \rho & 0 & 0 & 0 \\ v & 0 & \rho & 0 & 0 \\ (u^2+v^2)/2 & \rho u & \rho v & 1/(\gamma-1) & 0 \\ \tilde{\nu} & 0 & 0 & 0 & \rho \end{bmatrix}, \tag{15}$$

where γ is the specific heat ratio. We premultiply the time step term in equation 14 by a block diagonal $\mathbf{P}_t$ matrix made up of matrices from equation 15.

To accelerate convergence further we normalize the turbulence variable equation by $\sqrt{\tilde{\nu}_\infty}$. The turbulence variable becomes $\tilde{\nu}_{\text{norm}} = \tilde{\nu}/\sqrt{\tilde{\nu}_\infty}$ and we divide equation 1 by the same value,

$$\begin{aligned} &\partial_t(\rho\tilde{\nu}_{\text{norm}}) + \partial_j(\rho u_j \tilde{\nu}_{\text{norm}}) = \\ &\quad \partial_j\left(\frac{1}{\sigma}\rho(\nu+\nu')\partial_j\tilde{\nu}_{\text{norm}}\right) - \frac{1}{\sigma}(\nu+\nu')\partial_j\rho\partial_j\tilde{\nu}_{\text{norm}} + \sqrt{\tilde{\nu}_\infty}\frac{c_{b2}\rho}{\sigma}\partial_j\tilde{\nu}_{\text{norm}}\partial_j\tilde{\nu}_{\text{norm}} + \frac{P-D}{\sqrt{\tilde{\nu}_\infty}}. \end{aligned} \tag{16}$$

This scales the state variables to similar magnitudes to give the Newton solver a better conditioned system.

To converge to steady state, we linearize equation 14 about $\mathbf{u}^n$ with the preconditioner from equation 15,

$$\left(\frac{\mathbf{A}}{\Delta\mathbf{t}}\mathbf{P}_t + \left.\frac{\partial\mathbf{R}}{\partial\mathbf{u}}\right|_{\mathbf{u}^n}\right)\Delta\mathbf{u} = -\mathbf{R}\left(\mathbf{u}^n\right). \tag{17}$$

As the simulation proceeds, the time step is increased towards infinity and equation 17 recovers the full Newton step in equation 13.

Each nonlinear step requires solving the linear system in equation 17. To avoid forming the full residual Jacobian, we use the Generalized Minimum Residual method (GMRES) [21]. GMRES is a Krylov subspace method that solves the linear system $\mathbf{Ax} = \mathbf{b}$ by building a subspace through repeated multiplications of the $\mathbf{A}$ matrix. This process only requires a matrix vector product operator: given some vector $\mathbf{v}$ this operator outputs $\mathbf{Av}$. For our case, this operator can be constructed using Fréchet derivatives about the point $\mathbf{u}^n$,

$$\left.\frac{\partial\mathbf{R}}{\partial\mathbf{u}}\right|_{\mathbf{u}^n}\mathbf{v} \approx \frac{\mathbf{R}\left(\mathbf{u}^n+\epsilon\mathbf{v}\right)-\mathbf{R}\left(\mathbf{u}^n\right)}{\epsilon}. \tag{18}$$

We defined ϵ based on the work in [22, 20],

$$\epsilon = \begin{cases} e_{\text{rel}}\mathbf{v}\cdot\mathbf{u}^n/||\mathbf{v}||_2^2 & \text{if} \quad |\mathbf{v}\cdot\mathbf{u}^n| > e_{\min}||\mathbf{v}||_1 \\ e_{\text{rel}}e_{\min}\text{sign}(\mathbf{v}\cdot\mathbf{u}^n)||\mathbf{v}||_1/||\mathbf{v}||_2^2 & \text{otherwise,} \end{cases} \tag{19}$$

where $e_{\text{rel}} = 10^{-8}$ and $e_{\min} = 10^{-6}$.

A critical part of the success in GMRES is the strength of the preconditioner. The preconditioner is an approximate inverse of $\mathbf{A}$ that is applied as a matrix vector product to accelerate the convergence of GMRES. We used a block diagonal point Jacobi preconditioner, where each block corresponds to one cell. All the diagonal blocks of the residual Jacobian were computed analytically and then each block was inverted using an LU factorization with row pivoting. This preconditioner is recomputed for every nonlinear step. It provides adequate acceleration to run the small 2D test cases we are using to investigate the wall models. However, for production cases, something more complex is certainly needed.

The final piece of this puzzle is the nonlinear controller. The nonlinear controller modifies the time step in equation 17 based on the status of the nonlinear convergence. We modify the CFL to affect the computed time step. Figure 3 presents a flowchart of the nonlinear controller. Each nonlinear iteration starts by solving the linear system in equation 17. The initial $\Delta\mathbf{u}$ provided to the linear solver is $\mathbf{0}$. The linear solver is given a convergence criterion and a restriction on the number of Krylov subspace vectors it is allowed to build. The linear convergence criterion is a relative drop in the linear residual of four orders. As the steady solution is approached, this becomes a tighter absolute tolerance. This prevents the linear solver from spending too much time on linear solves when the nonlinear solver is still looking

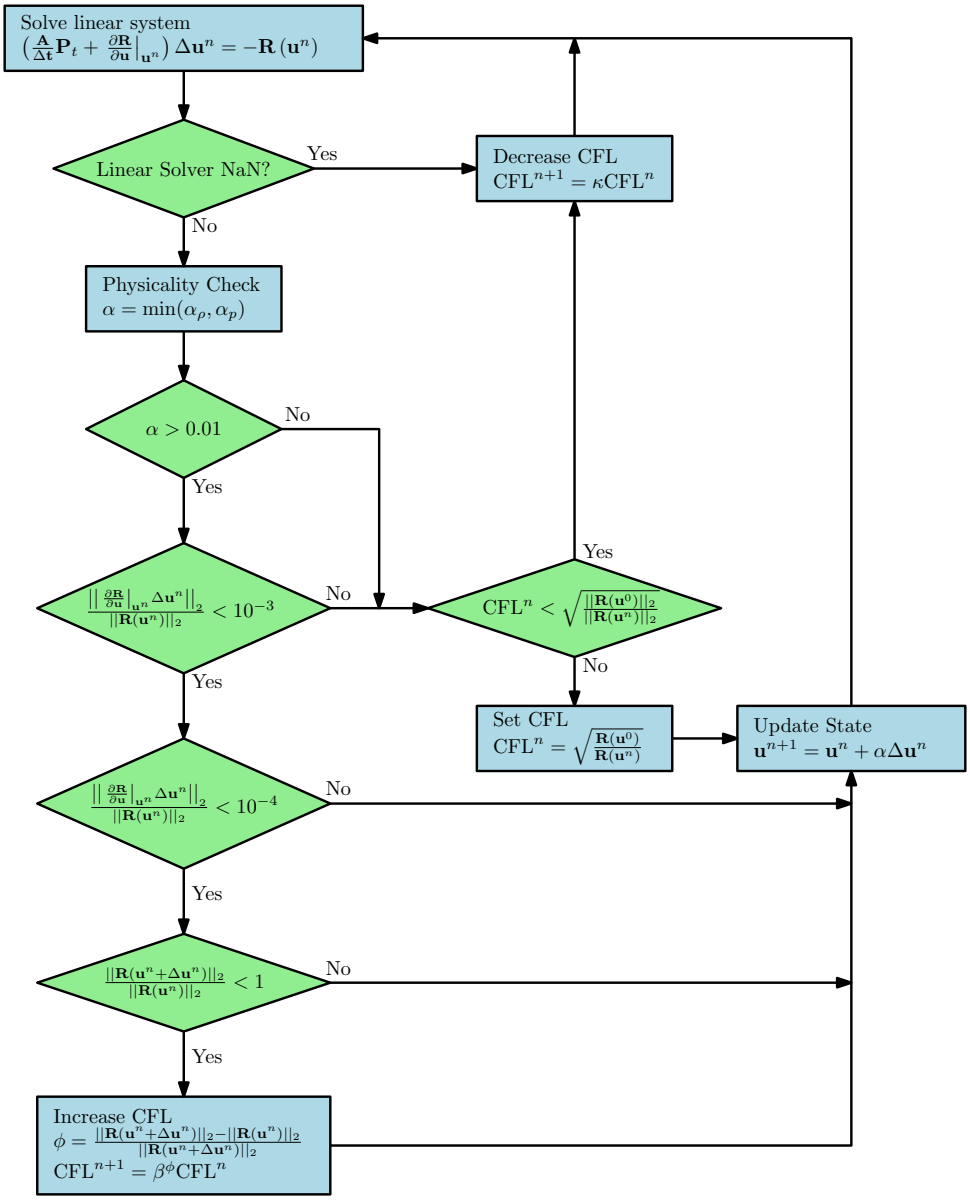


Figure 3: A flowchart describing the nonlinear controller logic. α is the fraction of the Newton step accepted; computed in equation 21. The CFL is increased with equation 25. The CFL is decreased with equation 23. We choose $\beta = 10$ and $\kappa = 0.5$ for the CFL controller parameters.

for the quadratic bucket.

After computing the linear solution, the nonlinear controller performs a line search to enforce a physicality constraint. Our line search is hard cut-off on the density and pressure values. For density, it is,

$$\alpha_\rho = \begin{cases} \frac{0.2-\rho^n}{\Delta\rho^n} & \text{if} \quad \rho^n + \Delta\rho^n < 0.2 \\ 1 & \text{otherwise,} \end{cases} \tag{20}$$

where α is the fraction of the full Newton step. We chose 0.2, but this could be any value greater than zero that the state should not decrease past. The same calculation is made for pressure and then the full system α is given by,

$$\alpha = \min(\alpha_\rho, \alpha_p). \tag{21}$$

If α is too small, the nonlinear controller attempts to reduce the CFL and continues to the next nonlinear iteration.

Whenever the nonlinear controller tries to reduce the CFL it checks if it is already below some minimum threshold,

$$\text{CFL}^n < \sqrt{\frac{||\mathbf{R}(\mathbf{u}^0)||_2}{||\mathbf{R}(\mathbf{u}^n)||_2}}. \tag{22}$$

If the CFL is already below this minimum threshold, the controller accepts the update and set the CFL to this minimum value. This is done to prevent the solver from stalling if it gets into a bad spot. Taking bad steps to get to a different spot in the design space is sometimes enough to obtain a solution. If the CFL is not below the threshold, it is reduced by,

$$\text{CFL}^{n+1} = \kappa \text{CFL}^n, \tag{23}$$

where we choose $\kappa = 0.5$. When the CFL is reduced the computed state update is rejected and the solver moves on to the next nonlinear iteration.

In addition to the relative convergence tolerance requested of the linear solver, there is a second relative converge tolerance specified by the nonlinear controller of 10^{-3}. If the linear system does not reach a relative convergence of 10^{-3} before maxing out the number of allowed Krylov subspace vectors, the nonlinear controller again attempts to reduce the CFL. If the linear system reached a relative convergence of 10^{-3}, but not 10^{-4}, the step is accepted and the CFL is left unchanged. If the linear system reaches a relative convergence of 10^{-4} *and* the steady-state nonlinear residual dropped after taking the update,

$$\frac{||\mathbf{R}(\mathbf{u}^n + \Delta\mathbf{u}^n)||_2}{||\mathbf{R}(\mathbf{u}^n)||_2} < 1, \tag{24}$$

the CFL is increased,

$$\text{CFL}^{n+1} = \beta^\phi \text{CFL}^n \quad \text{where} \quad \phi = \frac{||\mathbf{R}(\mathbf{u}^n + \Delta\mathbf{u}^n)||_2 - ||\mathbf{R}(\mathbf{u}^n)||_2}{||\mathbf{R}(\mathbf{u}^n + \Delta\mathbf{u}^n)||_2} \tag{25}$$

and we choose $\beta = 10$.

3.2 SA Wall Function

The SA wall function is a traditional wall function. The friction velocity, u_τ, in equation 2 is fit to the forcing point data using a Newton method. Once u_τ is computed, equation 2 provides u and $\partial u/\partial\eta$ as a function of η.

3.3 ODE Wall Function

The ODE wall function requires solving a 1D ODE instead of fitting a single parameter. We implemented a high-order continuous Galerkin (CG) method with Mesh Optimization via Error Sampling and Synthesis (MOESS) for solving equation 4. The convection terms in equation 4 are constants with respect to our state variables, giving us an ODE with no advection terms. This lends itself to a CG formulation. The solver uses a Newton–Krylov solver with pseudo-transient continuation to drive the residuals to zero. Unlike the background flow solver, the ODE solver builds the full residual Jacobian for performing

matrix vector products in GMRES. Using the notation in equation 11 we can rewrite equation 4,

$$\mathbf{F}(\mathbf{u}) = \begin{bmatrix} -(\mu + \mu_t)\frac{\partial u}{\partial y} \\ -(\nu + \tilde{\nu})\frac{\partial \tilde{\nu}}{\partial y} \end{bmatrix}, \tag{26}$$

and,

$$\mathbf{S}(\mathbf{u}) = \begin{bmatrix} \left.\frac{\partial p}{\partial x}\right|_F + \psi(y)\ \rho|_F \left[u_F\ \left.\frac{\partial u}{\partial x}\right|_F + v_F\ \left.\frac{\partial u}{\partial y}\right|_F\right] \\ -c_{b2}\left(\frac{\partial \tilde{\nu}}{\partial y}\right)^2 - \sigma(\text{production} - \text{destruction}) \end{bmatrix}. \tag{27}$$

Since an ODE is solved in every cut cell during every residual evaluation, it is critical that the ODE solver is both fast and robust. To facilitate speed, a Block ILU(0) preconditioner is used to accelerate the GMRES convergence. A block consists of the 2 by 2 matrix associated with each basis function rather than the traditional element block present in discontinuous Galerkin methods. No reordering of the rows is necessary as this is a 1D problem. This preconditioner allows GMRES to solve most systems with hundreds of variables in less than ten Krylov subspace vector iterations. To facilitate robustness, the GMRES method is allowed to build the full subspace if necessary to ensure a solution is found.

The general nonlinear strategy remains consistent with that of the background solver with special considerations for robustness. The physicality check for this solver requires that $\tilde{\nu} > 0$. When the solver wants to take a step that causes one of the turbulence variable states to go negative, the CFL is reduced and the linear step is rejected. If the CFL is at the lower allowable bound, the computed step is taken for all basis functions except the non-physical one. In general, this allows the problematic basis function to recover. We do not check for negative turbulence variables everywhere in the cell, only at the basis function locations. We found this sufficient to obtaining converged, physical, solutions.

Even with a tuned nonlinear controller, it is still possible for the ODE solution to fail. When this happens, we reduce the complexity of the ODE for the rest of the global flow's current nonlinear step. The x-momentum source term in equation 4 is removed. This is sufficient to protect the ODE solver except on particularly coarse meshes. Even without the convection source term, the ODE is theoretically more accurate than the SA wall function because it allows for nonlinear variation in the turbulence variable. In practice this difference is minuscule compared to the change once the convection source term is added. As a last resort, the ODE is discarded entirely for the analytical SA wall function. After each nonlinear iteration the complexity of each ODE is increased by one level if it is reduced. Allowing the ODE to reintroduce complexity during linear solves was tested, but prevented linear convergence in most cases where it was needed.

To ensure accurate solutions for a wide variety of boundary conditions, we implemented MOESS in conjunction with our CG solver [23]. MOESS iteratively determines the optimal change in a metric field given a prescribed metric-cost and metric-error relationship. We use it to optimally distribute a desired number of degrees of freedom along our domain to get an accurate wall shear stress.

To use MOESS, an element error indicator is required. We use the adjoint weighted residual,

$$\delta J \approx -\mathbf{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H), \tag{28}$$

where δJ is the error of some functional J with respect to each state, $\mathbf{\Psi}_h$ is the fine space adjoint, and $\mathbf{R}_h\left(\mathbf{U}_h^H\right)$ is the fine space residual evaluated at the coarse space solution prolonged to the fine space. We chose an increase in the solution order, p, as our fine space and used the wall shear stress as J. We solved the fine space adjoint using GMRES with an exact residual Jacobian. The error indicator for an element is computed by summing over the error contribution for each of its basis functions. The basis functions that are shared between elements contribute half of their error to each element. MOESS uses this error indicator to update a Riemann metric field.

A Riemann metric field, $\mathcal{M}(\vec{x})$, is a field of symmetric positive definite (SPD) tensors that encode information about the desired mesh size. In multiple dimension, it can encode stretching and rotations to generate anisotropic meshes. In one dimension it is a scalar field the indicates the desired size of elements. The metric provides a way to measure the distance from a point, $\vec{x}$, to another point infinitesimally far away, $\vec{x} + \delta\vec{x}$. This distance is,

$$\delta l = \sqrt{\delta\vec{x}^T \mathcal{M} \delta\vec{x}}, \tag{29}$$

and in 1D,

$$\delta l = \delta x \sqrt{\mathcal{M}}. \tag{30}$$

Every element should be of length one under the metric.

Given a metric field, we compute a new mesh by marching unit distances along our domain until we reach the end. MOESS provides a metric field that corresponds to a mesh with a user specified, target, number of degrees of freedom (dof). This metric field is stored at the previous mesh nodes. We assume the logarithm of the metric varies linearly between two nodes [24],

$$M_x = M_a \left(\frac{M_b}{M_a} \right)^{\frac{x-a}{b-a}}, \tag{31}$$

where M_a is the metric at the vertex $a < x$ and M_b is the metric at the vertex $b > x$. To compute the metric length between two points, a and b, we use,

$$L_e = \begin{cases} \frac{L_a - L_b}{\log L_a / L_b} & |L_a - L_b| > 0.001 \\ \frac{L_a + L_b}{2} & \text{else} \end{cases}, \quad \text{with} \quad L_a = l_e \sqrt{M_a}, L_b = l_e \sqrt{M_b}. \tag{32}$$

For 1D CG, the total number of degrees of freedom (dof) is $Np + 1$, where N is the number of elements and p is the solution order. We can compute the appropriate number of elements for the new mesh using the requested number of degrees of freedom. From the definition of the metric field, each element should be the same length under the metric. Ideally, MOESS computes a metric field such that each element is length one under the metric. This is not always the case, so we use equation 32 to compute the piecewise integral of the metric over the entire domain. This is divided by the number of elements requested to get the constant valued metric length for all the elements. To generate the mesh, we start at $y = 0$ and place points at the constant valued metric length distances apart until the other end of the domain is reached.

Equation 32 in conjunction with a Newton solver is used to compute the location of each point in the new mesh. Each point in the new mesh will lie between two points on the background mesh, $[a, b]$. To place that point, the interval $[a, b]$ is determined by checking the integral of the metric from the previously placed new mesh point and the next old mesh point. If that distance does not reach the desired metric length, then the next interval defined by the old mesh point and the next old mesh point is checked. This proceeds until the correct interval $[a, b]$ is found. Then, equation 32 is cast as a residual and a Newton solver identifies x such that the metric length between the previous new mesh point and this point is the desired length.

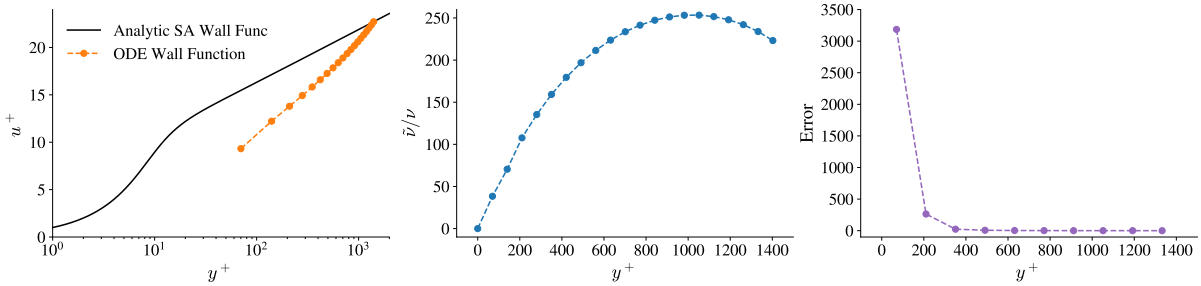
Figure 4 shows the progression of MOESS on a $p = 2$ solution of the ODE solution starting with a uniformly spaced mesh. The boundary conditions for this case comes from the 2D turbulent bump case on the NASA turbulence modeling resource website [25]. The finest CFL3D solution was used to obtain forcing point data downstream of the bump. Each MOESS cycle consists of a flow solution, followed by the MOESS mesh optimization. Figure 4a indicates that the naive uniform mesh is terrible. The velocity profile does not come close to matching what we expect. Even with this mesh, the Newton–Krylov solver is robust enough to easily converge the system. After the first adaptation, the number of cells is doubled, and the mesh points start sliding towards the wall. The velocity profile converges to what we would expect. This forcing point is approaching the wake region, and the ODE is able to begin predicting this transition.

4 Results

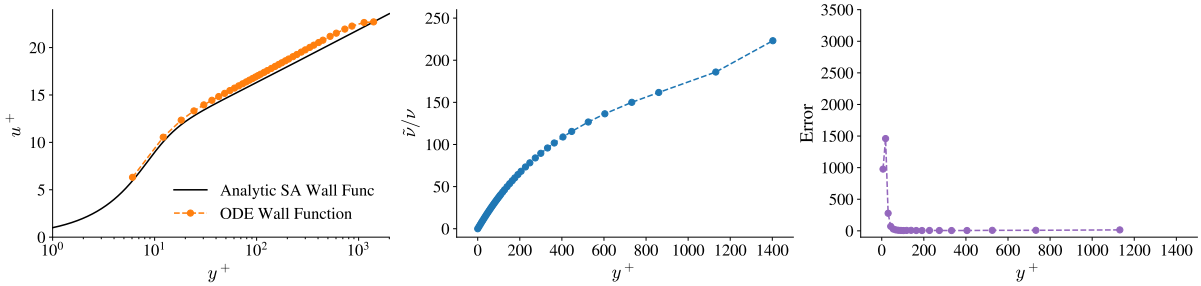
4.1 Turbulent Finite Flat Plate

The turbulent flat plate case tests the adiabatic wall boundary conditions without introducing numerical difficulties around cut-cells. This case is run at Mach 0.2 with a Reynolds number 10^7 and contains a plate that extends ten reference lengths. For the WMRANS we set $\chi_{t0} = 20$. We ran a sequence of meshes for this case. The L0 mesh had an initial off-wall spacing of $y \approx 10^{-4}$. Each subsequent mesh performed isotropic coarsening to double the initial off-wall spacing. The far-field retains the same mesh size, so each level is not an exact 25% decrease in the number of cells. The L0 mesh has approximately $650k$ isotropic cells. The L3 mesh has approximately $84k$ isotropic cells.

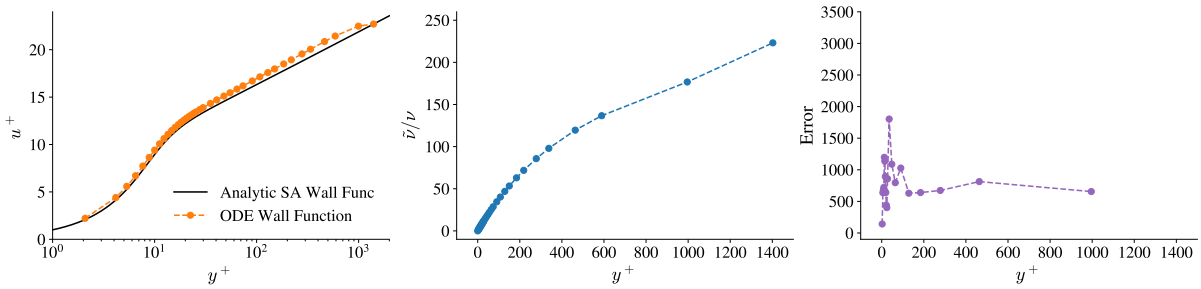
Figure 5 presents velocity profiles for all three methods at three different stations. The two wall



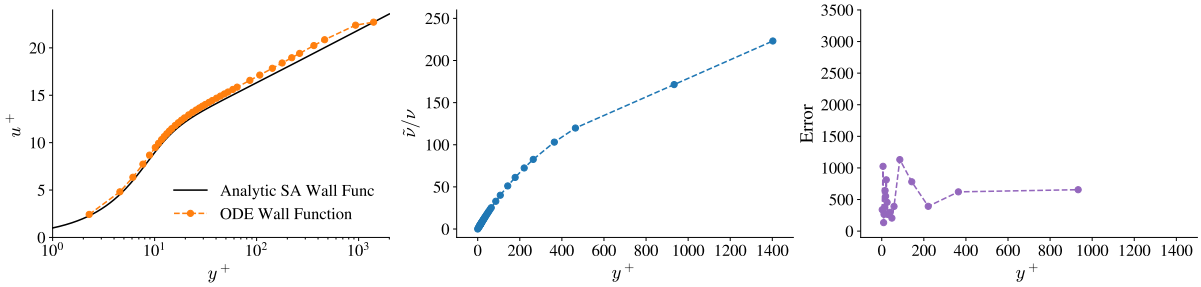
(a) Initial mesh with 10 linearly spaced $p = 2$ elements.



(b) Mesh after three MOESS cycles with 20 elements.

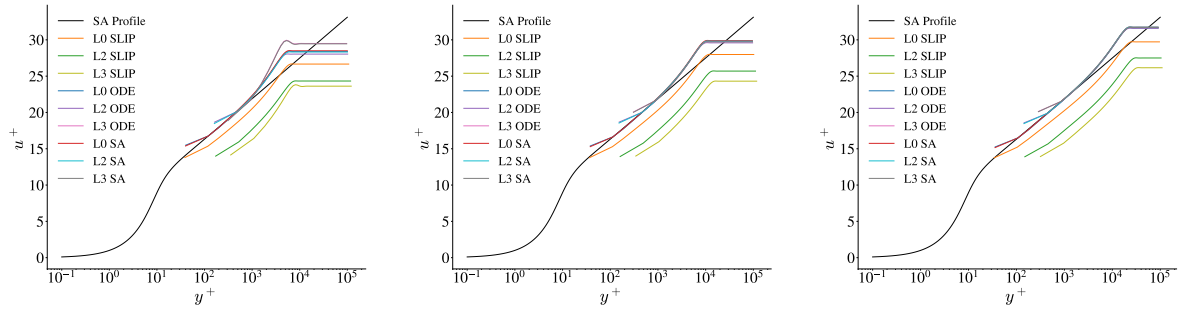


(c) Mesh after six MOESS cycles with 20 elements.



(d) Mesh after nine MOESS cycles with 20 elements.

Figure 4: MOESS sequence on the ODE with $p = 2$ elements. The initial mesh is ten linearly spaced elements. The target number of elements is twenty. These meshes are adapted on the gradient of the velocity at $y = 0$. A nodal basis is used, so the mesh points correspond to the basis function locations. The u^+ and $\tilde{\nu}$ plots show points at each basis function's global space location. The error plots show points at the middle of each element. The analytic SA wall function shown is the one used in the x-momentum source in equation 4. It is not necessarily accurate given the forcing point location.



(a) Velocity profiles at $Re_x = 1 \times 10^7$ (b) Velocity profiles at $Re_x = 2 \times 10^7$ (c) Velocity profiles at $Re_x = 5 \times 10^7$

Figure 5: Velocity profiles for the zero pressure gradient finite flat plate case. The plate is ten reference lengths long. This was run at Mach 0.2 and Reynolds number 10^7. The L0 mesh has initial off-wall spacing at $y \approx 10^{-4}$. Each subsequent level of mesh doubles the off-wall spacing.

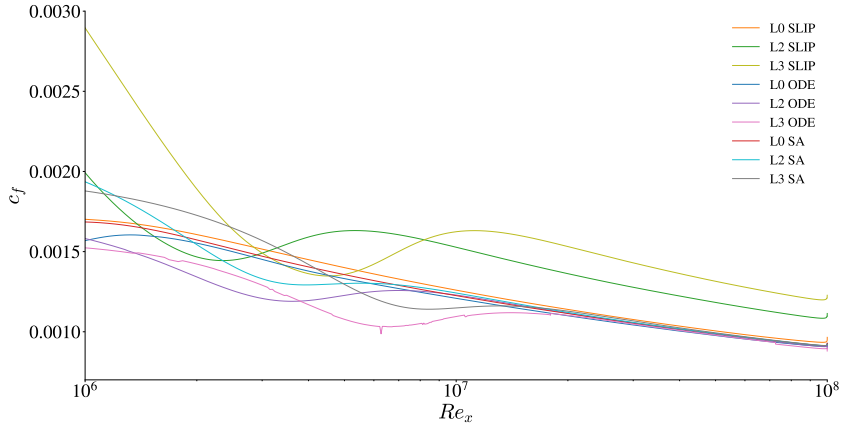


Figure 6: The skin friction coefficient for the finite flat plate case. The L0 mesh has an initial off-wall spacing at $y \approx 10^{-4}$. Each subsequent level of mesh doubles the off-wall spacing.

functions match the expected profile well at each station. In both cases, there is a single point that is not on the profile. This is the cut cell point, the first point on the line is the forcing point cell. The cut cell does not provide the states necessary to compute skin frictions, since this information comes from the forcing point. The WMRANS solution is converging to the appropriate solution as the mesh is refined. We only see the last point in each velocity profile begin to converge to the expected linear velocity solution for the WMRANS. This implies that the WMRANS needs additional resolution than the wall function approaches even though the profile is more benign.

Figures 6 and 7 show the skin frictions and pressure coefficients respectively. The biggest difference in between the mesh levels is seen in the skin friction plot. The x-axis is on a log scale, making it clear that the biggest discrepancy is the boundary layer start up. The wall transverse direction contains constant mesh resolution for a given mesh. We see as the mesh is refined, the skin friction converges to the asymptotic value sooner. In the coarsest ODE mesh result, we can see discontinuities in the skin friction where the wall function has had to change the complexity of the wall function to converge the ODE. Again, we see the same behavior with the WMRANS approaching the expected solution as the mesh resolution is increased.

Figure 8 compares the previously used Runge–Kutta scheme developed [17] with the new Newton–Krylov scheme. Even with a relatively weak precondtioner, the Newton–Krylov scheme is easily able to outperform the Runge-Kutta scheme. In most cases, we found that once we found the Newton bucket, the convergence rate was inhibited by the linear solver's ability to solve increasingly stiff systems as the CFL ramped, limiting nonlinear convergence in the bucket to steep linear convergence.

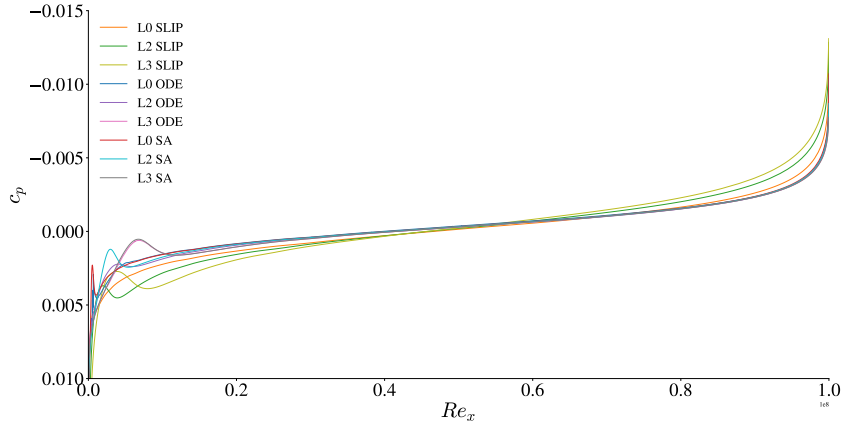


Figure 7: The pressure coefficients for the finite flat plate case. The L0 mesh has an initial off-wall spacing at $y \approx 10^{-4}$. Each subsequent level of mesh doubles the off-wall spacing.

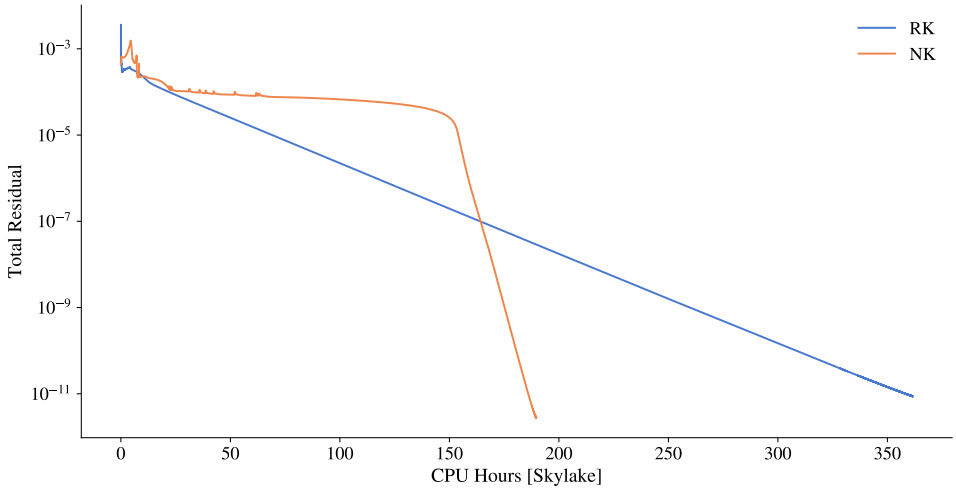
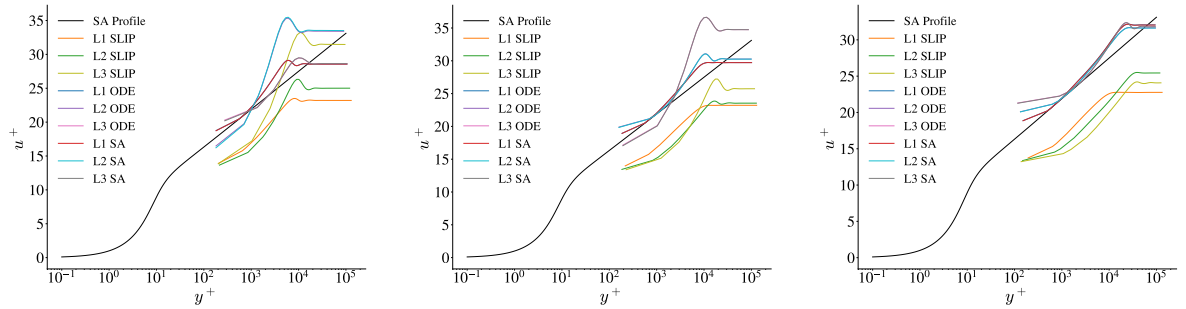


Figure 8: The time to converge the turbulent flat plate case on the L3 mesh.



(a) Velocity profiles at $Re_x = 1 \times 10^7$ (b) Velocity profiles at $Re_x = 2 \times 10^7$ (c) Velocity profiles at $Re_x = 5 \times 10^7$

Figure 9: Velocity profiles for the zero pressure gradient finite flat plate case with cut-cells. The plate is ten reference lengths long. This was run at Mach 0.2 and Reynolds number 10^7. The L0 mesh has initial off-wall spacing at $y \approx 10^{-4}$. Each subsequent level of mesh doubles the off-wall spacing.

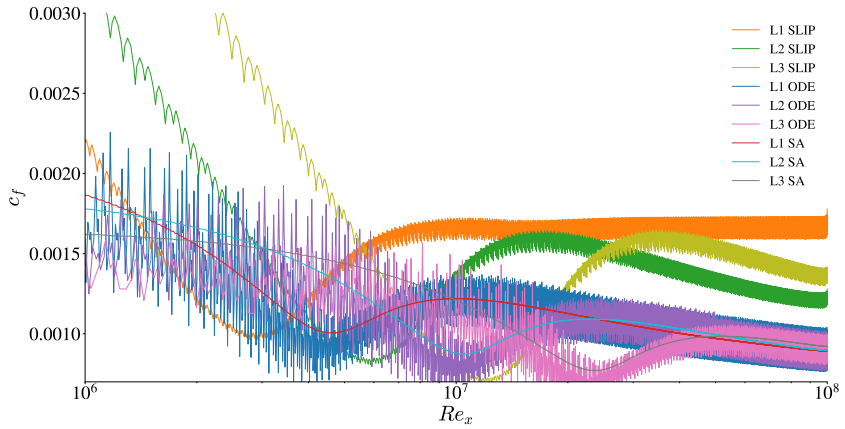


Figure 10: The skin friction for the rotated finite flat plate case. The L0 mesh has an initial off-wall spacing at $y \approx 10^{-4}$. Each subsequent level of mesh doubles the off-wall spacing.

4.2 Rotated Finite Flat Plate

This is the same case as the rotated finite flat plate, but with cut-cells introduced. Rather than placing the flat plate along the bottom boundary of the domain, it is set into the domain at a 15°angle (figure 11). We used the same conditions as the wall-aligned flat plate (Mach 0.2 and Reynolds number 10^7).

Figure 9 shows the velocity profiles for the rotated finite flat plate. The solver struggles considerably more when cut-cells are introduced. The gradient stencils in these cells are degraded and the principle directions are no longer aligned with the mesh. We still see similar behavior in the convergence of the results, as the mesh is refined all solutions look to approach the expected results.

Figure 10 shows the skin friction coefficients for the rotated flat plate case. The skin frictions are considerably more noisy for the ODE wall function and WMRANS. The pressure gradients in figure 11 begin to reveal why this might be. This is a zero pressure gradient case, and we would expect to see no pressure gradients. Small patterns in the mesh affect the pressure, resulting in an oscillating pressure gradient that follows the triangle cut-cell pattern along the wall. The oscillations only appear around the cut cells, and as the mesh is refined, the physical location of the oscillations moves closer to the wall. This behavior is independent of the method we choose to resolve the wall, even with the analytical SA function. These oscillations do not appear in the density, velocity, or turbulence variable states. However, the oscillations do show up in the internal energy, implying that the problem is localized to the pressure. We do not see this behavior when running inviscid cases (see figure 13). It would appear that the current viscous flux scheme is not adequate for resolving viscous fluxes in and around cut cells. To alleviate some of this issue, we implemented a bilinear interpolation for the pressure gradient at the forcing points based on the face neighbors of the forcing point cells.

The SA wall function is able to hide this in the skin friction because the only information it uses at the forcing point is the tangential velocity momentum, which does not oscillate. The ODE wall function

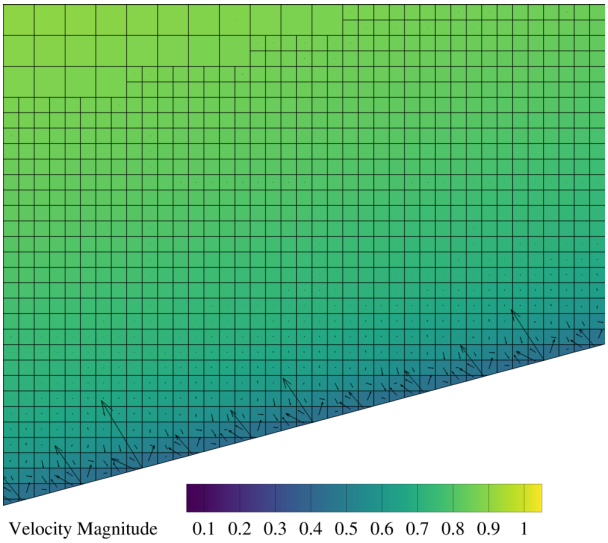
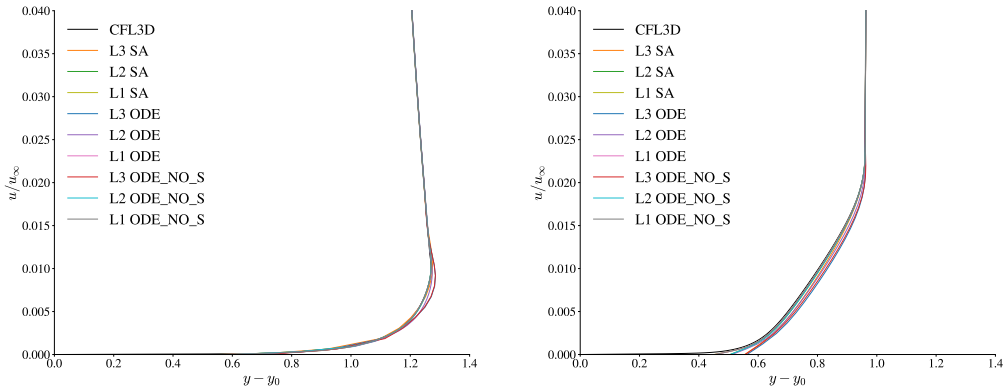


Figure 11: A plot of the velocity magnitude and pressure gradients on the rotated flat plate in the boundary layer. The vectors are the computed least-squares gradients used in the inviscid fluxes, viscous fluxes, and wall function forcing points. The pressure gradient shows regular non-physical oscillations. These oscillations are a mesh artifact centered around triangle cut cells.



(a) $x = 0.75$ (b) $x = 1.20148$

Figure 12: The velocity profiles for the 2D bump case.

contains $\partial p/\partial \xi$ in the x-momentum source term and is directly affected by the pressure oscillations. All of the information it uses to compute viscous fluxes on cut-cell faces then propagates these oscillations into the fluxes. WMRANS also depends more on the states in and around cut-cells than the SA wall function because there is no notion of a wall function to improve the states used at faces.

Even with the oscillations, the ODE solution is able to get something that resembles expected behavior. The slip wall struggles with these oscillations, particularly in converging solutions on the finer grids. We expect that with more accurate viscous fluxes in the cut cells that WMRANS would perform similar to the grid aligned case.

4.3 2D Bump

We ran the 2D bump-in-channel verification case from the NASA turbulence modeling resource [25]. It was run at Mach 0.2 and Reynolds number 3×10^6. The L1 mesh has approximately $220k$ isotropic cells. The L3 mesh has approximately $59k$ isotropic cells. This case introduces a pressure gradient into the solution.

Figure 12 shows two velocity profiles on the top of the bump and in the wake region. In all cases, the mesh converged solution matches the CFL3D solution. We introduce a 'new' wall function that is the same as the ODE, but without the x-momentum source terms in equation 4.

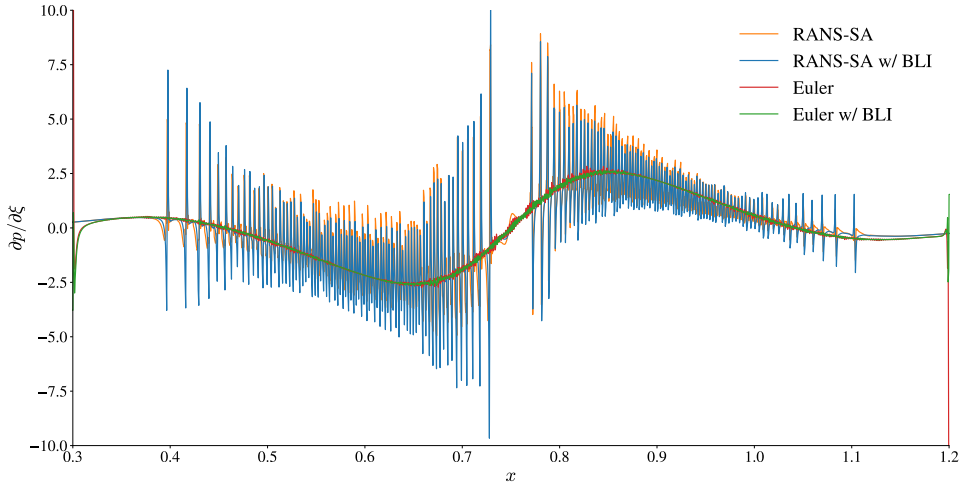


Figure 13: The wall tangential pressure gradient used at the forcing point for the ODE wall function. The bilinear interpolation solutions are smoother than the solutions without the interpolation. We used an Euler pressure profile to test a more realistic pressure gradient profile.

For the bump case, we verified the bilinear interpolation of the pressure gradient using an inviscid solution. Figure 13 shows the effect of the bilinear interpolation on an inviscid and a viscous case. The interpolation has a smoothing effect in both cases, but it is not enough to recover an acceptable pressure gradient in the viscous case.

Figure 14 presents the pressure coefficients along the bump. Both the SA wall function and the ODE wall function are able to predict the correct pressures on all of the mesh levels. There are spurious pressure spikes around triangle cut cells. These spikes are reduced as the mesh resolution is increased.

Figure 15 presents the skin frictions along the bump. The SA analytical wall function matches the CFL3D solution well even though it assumes no pressure gradient. As the mesh resolution is increased the skin friction approaches the expected profile. The ODE wall function follows the general trend but is noisy. Removing the x-momentum source term from the ODE smooths these skin frictions. This reinforces the assessment that the wall function methods with no pressure dependence are able to generate smooth skin frictions.

5 Conclusion

In this paper we developed a Newton–Krylov solver for Viscous Aerodynamic Cartesian Cut cells (VACC). This Newton solver allowed us to prototype new wall function and wall model scheme for solving the high Reynolds number RANS-SA equations. We tested three separate wall functions and wall model developed by Ursachi et al. [16].

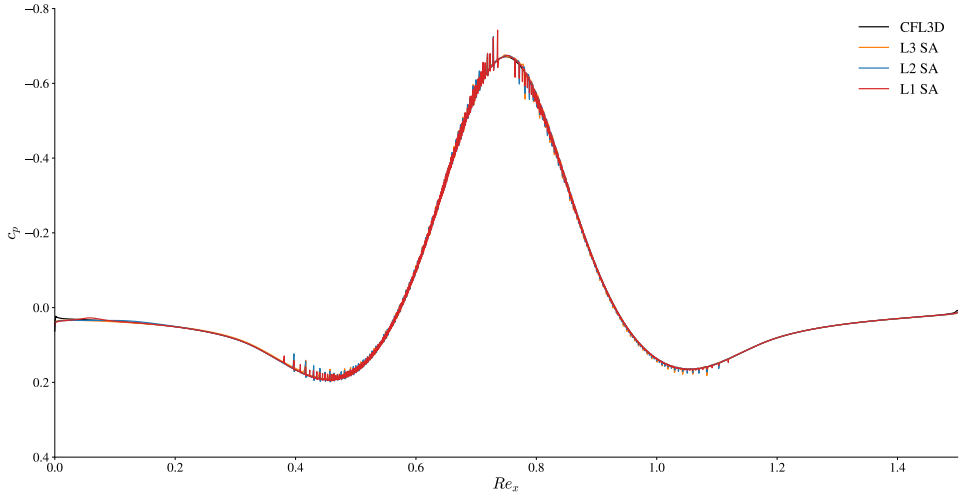
Each approach showed promise when dealing with the rapidly changing velocity profile near the wall. However, the interior viscous flux routine common to all methods broke down in the cut cells. Before providing a final assessment between all of the methods, the interior viscous flux routine needs to be reassessed.

The SA analytical wall function is able to hide the oscillations in the pressure because it only depends on the x-momentum at the forcing point. WMRANS and the ODE wall function both suffer from the oscillating pressure resulting in noisy skin frictions. Future work will be focused on addressing the issues present in this viscous flux scheme in and around cut cells.

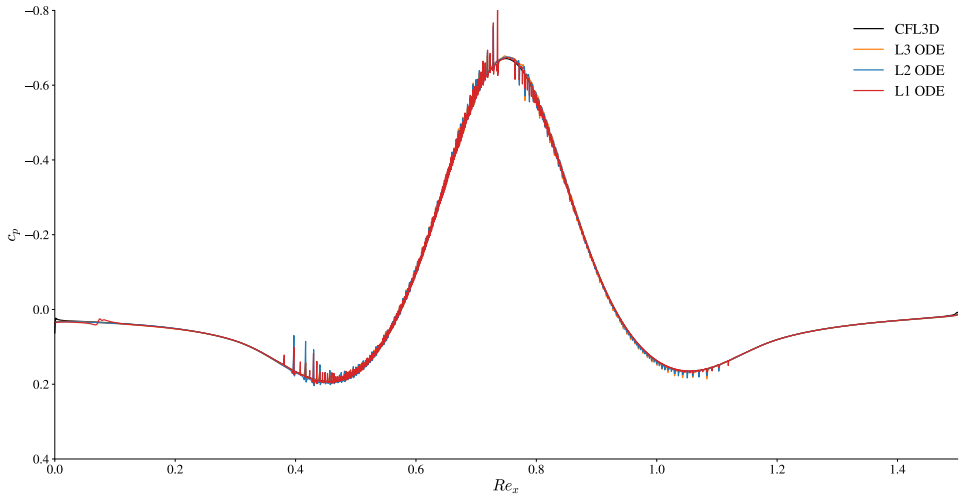
References

[1] A. Jameson, L. Martinelli, and N. A. Pierce. Optimum aerodynamic design using the Navier–Stokes equations. *Theoretical and Computational Fluid Dynamics*, 10(1–4):213–237, 1998.

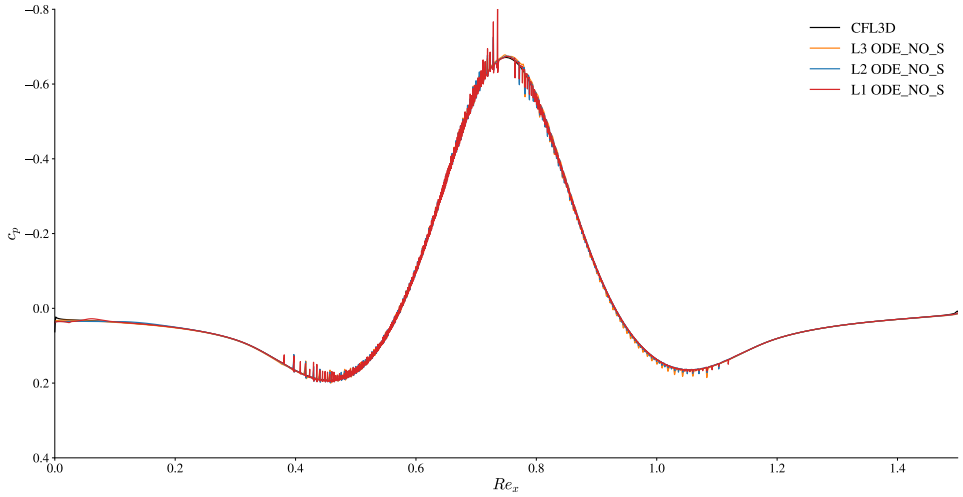
[2] W. K. Anderson and V. Venkatakrishnan. Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation. *Computers and Fluids*, 28(4):443–480, 1999.



(a) SA Analytical Wall Function

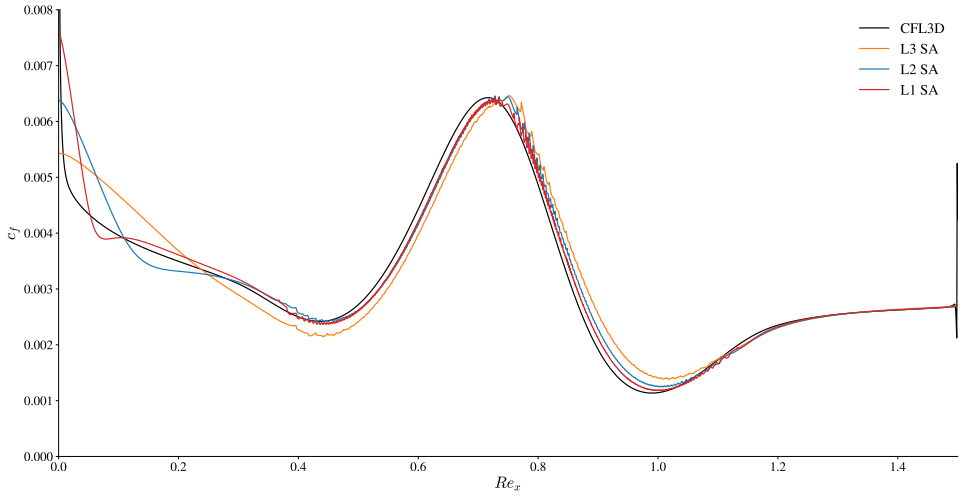


(b) ODE Wall Function

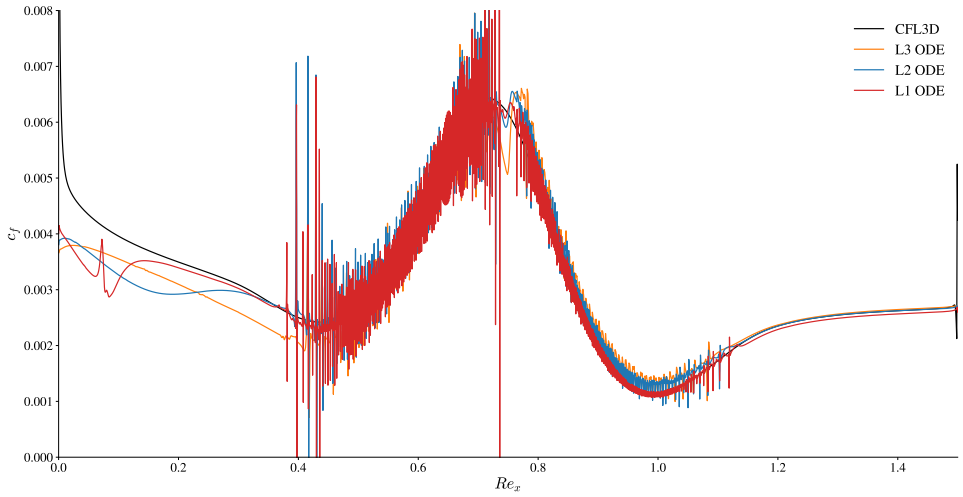


(c) ODE Wall Function with no x-momentum source terms

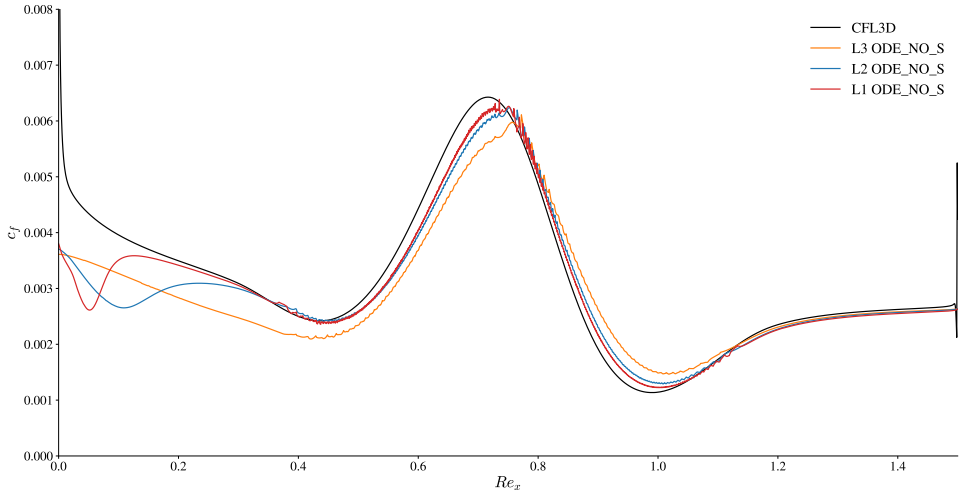
Figure 14: The pressure coefficients for the 2D bump case. The pressure matches the CFL3D solution well on all levels of meshes. However, the pressure oscillations are evident around this solution.



(a) SA Analytical Wall Function



(b) ODE Wall Function



(c) ODE Wall Function with no x-momentum source terms

Figure 15: The skin frictions for the 2D bump case. The pressure follows the CFL3D solutions well. Each wall function exhibits convergence towards the expected solution as the mesh is refined. The effect of the pressure oscillations in the solution are clear in the ODE. Removing the x-momentum source term smooths the ODE result.

[3] Eric J. Nielsen and W. Kyle Anderson. Aerodynamic design optimization on unstructured meshes using the Navier–Stokes equations. *AIAA Journal*, 37(11):1411–1419, 1999.

[4] A. Jameson, L. Martinelli, J. J. Alonso, J. C. Vassberg, and J. Reuther. *Computational Fluid Dynamics for the 21st Century*, chapter Perspectives on Simulation Based Aerodynamic Design, pages 135–178. Springer, Berlin, Heidelberg, 2001.

[5] Joaquim R. R. A. Martins. Aerodynamic design optimization: Challenges and perspectives. *Computers & Fluids*, 239:105391, May 2022.

[6] J. Slotnick, A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp, E. Lurie, and D. Mavriplis. CFD vision 2030 study: A path to revolutionary computational aerosciences. Technical Report CR–2014-218178, NASA, March 2014.

[7] Michael J. Aftosmis. Lecture notes in solution adaptive Cartesian grid methods for aerodynamic flows with complex geometries. *VKI Lecture Series*, 1997.

[8] Michael J. Aftosmis, Marsha J. Berger, and John E. Melton. Robust and efficient Cartesian mesh generation for component-based geometry. *AIAA Journal*, 36(6), June 1998.

[9] William J Coirier and Kenneth G Powell. Solution-adaptive Cartesian cell approach for viscous and inviscid flows. *AIAA Journal*, 34(5):938–945, 1996.

[10] Hongwu Zhao, Patrick Hu, Ramji Kamakoti, Nagendra Dittakavi, Michael Aftosmis, David Marshall, Liping Xue, Kan Ni, and Shaolin Mao. Towards efficient viscous modeling based on Cartesian methods for automated flow simulation. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, page 1472, 2010.

[11] Tao Ye, Rajat Mittal, HS Udaykumar, and Wei Shyy. An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries. *Journal of computational physics*, 156(2):209–240, 1999.

[12] Marsha Berger and Michael Aftosmis. Progress towards a Cartesian cut-cell method for viscous compressible flow. In *50th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, page 1301, 2012.

[13] Francesco Capizzano. Turbulent wall model for immersed boundary methods. *AIAA Journal*, 49(11):2367–2381, 2011.

[14] Alex Kleb, Krzysztof J. Fidkowski, and Joaquim R. R. A. Martins. Development of a Cartesian cut-cell solver for viscous flows. In *AIAA SciTech Forum*, January 2023.

[15] Marsha J Berger and Michael J Aftosmis. An ODE-based wall model for turbulent flow simulations. *AIAA Journal*, 56(2):700–714, 2018.

[16] Carmen-Ioana Ursachi, Steven R Allmaras, David L Darmofal, and Marshall C Galbraith. Stress-equivalent Spalart–Allmaras wall model with local boundary conditions for reynolds-averaged navier–stokes. *AIAA Journal*, pages 1–17, 2024.

[17] Alex Kleb, Krzysztof J. Fidkowski, and Joaquim R. R. A. Martins. Extension of a viscous cartesian cut-cell solver to the compressible RANS equations. In *AIAA Aviation Forum*, San Diego, CA, June 2023.

[18] Marco Ceze and Krzysztof Fidkowski. Pseudo-transient continuation, solution update methods, and CFL strategies for DG discretizations of the RANS-SA equations. In *21st AIAA Computational Fluid Dynamics Conference*, Fluid Dynamics and Co-located Conferences. American Institute of Aeronautics and Astronautics, June 2013.

[19] Mohagna J Pandya, Boris Diskin, James L Thomas, and Neal T Frink. Improved convergence and robustness of USM3D solutions on mixed-element grids. *AIAA Journal*, 54(9):2589–2610, 2016.

[20] Anil Yildirim, Gaetan K. W. Kenway, Charles A. Mader, and Joaquim R. R. A. Martins. A Jacobian-free approximate Newton–Krylov startup strategy for RANS simulations. *Journal of Computational Physics*, 397:108741, November 2019.

[21] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, January 2003.

[22] Peter N. Brown and Youcef Saad. Hybrid krylov methods for nonlinear systems of equations. *SIAM Journal on Scientific and Statistical Computing*, 11(3):450–481, May 1990.

[23] Masayuki Yano and David L Darmofal. An optimization-based framework for anisotropic simplex mesh adaptation. *Journal of Computational Physics*, 231(22):7626–7649, 2012.

[24] Daniel Ibanez, Nicolas Barral, Joshua Krakos, Adrien Loseille, Todd Michal, and Mike Park. First benchmark of the unstructured grid adaptation working group. *Procedia engineering*, 203:154–166, 2017.

[25] Christopher Rumsey. NASA turbulence modeling resource. `https://turbmodels.larc.nasa.gov`, 2019. Accessed: 2019-03-27.