Computers and Fluids 304 (2026) 106882

Contents lists available at ScienceDirect

Computers and Fluids

journal homepage: www.elsevier.com/locate/compfluid

Solving high Reynolds number flows on Cartesian cut-cell meshes using an ODE wall function with momentum balance

Alex Kleb *, Krzysztof J. Fidkowski, Joaquim R. R. A. Martins

Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI, 48109, USA

ARTICLE INFO

Keywords:
Cartesian cut cells
Computational fluid dynamics
Turbulence modeling

ABSTRACT

Computational fluid dynamics is essential for designing aircraft, turbines, and other engineering systems. However, generating suitable computational meshes for complex geometries remains the primary bottleneck in analysis workflows, often requiring days of manual effort. Traditional boundary-conforming meshes excel at capturing near-wall physics in viscous flows but demand specialized expertise and extensive preprocessing time. Cartesian cut-cell methods provide automatic mesh generation for complex geometries in minutes, yet they struggle with high Reynolds number viscous flows where boundary layers exhibit rapid velocity changes that require prohibitively fine resolution for isotropic elements. The fundamental challenge is accurately modeling boundary layer physics on automatically generated meshes without sacrificing the computational efficiency that makes such methods attractive. In this work, we show that an ordinary differential equation (ODE) wall function incorporating pressure-momentum balance enables accurate high Reynolds number viscous flow predictions on coarse Cartesian cut-cell meshes. Our approach solves a one-dimensional boundary value problem at each wall boundary face that accounts for the transition from the viscous dominated near-wall region to the inviscid wake region, allowing forcing points to operate effectively at $y^+ > 600$. Unlike traditional wall functions, the ODE is not limited to the logarithmic layer and maintains accuracy in strong pressure gradient environments typical of aerodynamic applications. The ODE can achieve correct skin friction predictions on meshes more than four times coarser than analytical wall functions require. The ODE wall functions are solved with a robust Newton–Krylov implementation that utilizes adaptive mesh refinement. It converges reliably across diverse flow conditions while solving hundreds of degrees of freedom in fewer than ten linear iterations. These results demonstrate that automatic high-fidelity viscous flow analysis is achievable without manual mesh generation expertise.

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) [1–4].

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 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 [9–12]. 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 wall-normal direction than the

* Corresponding author.
E-mail address: akleb@umich.edu (A. Kleb).

https://doi.org/10.1016/j.compfluid.2025.106882
Received 31 March 2025; Received in revised form 9 October 2025; Accepted 17 October 2025
Available online 22 October 2025

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 unnecessary 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 obtain wall-tangential velocities and gradients to augment flux computations [14].

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 [12,15]. 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, 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 [16] developed a unique type of wall function that solves a one-dimensional ordinary differential equation (ODE) instead of 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 pressure-momentum balance in the wake region, allowing the model to remain valid even further from the wall than the SA wall function. It also allows the model to better predict the boundary layer when strong pressure gradients are present. However, implementing a complex wall function like this can lead to issues. The ODE solver needs to retain the accuracy necessary to resolve correct profile, while also remaining computationally efficient. Determining the correct balance and ensure that the global nonlinear solver is able to converge reliably remains an issue.

Ursachi et al. [17] 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 undergoes 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. A similar approach has been applied to Cartesian immersed boundary methods by Tamaki et al. [18].

The objective of this research is to investigate and compare different near-wall approaches for modeling high Reynolds number flows on Cartesian cut-cell meshes. In this paper, we develop a numerical framework with the ODE wall function described by Berger and Aftosmis [16] and integrate it into a global Newton–Krylov solver. Then, 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, the way each boundary method fits in, and a novel implementation of the ODE wall model that incorporates a 1D mesh optimization algorithm. Four exterior flow cases are presented in Section 4 to verify the methods and compare their strengths and weakness. Finally, we end with concluding remarks in Section 5.

2. Governing equations

We focus on solving the steady 2D compressible Reynolds-averaged Navier–Stokes (RANS) equations,

$$\partial_j(\rho) + \partial_j(\rho v_j) = 0, \tag{1}$$

$$\partial_j(\rho v_j) + \partial_j(\rho v_j v_i + \delta_{ij} p) - \partial_j \tau_{ij} = 0, \tag{2}$$

$$\partial_j(\rho E) + \partial_j(\rho H v_j) - \partial_j(v_i \tau_{ij} - Q_i) = 0, \tag{3}$$

where ρ is the fluid density, v_i is the ith component of the velocity vector, $\vec{v} \in \mathbf{R}^d$, δ_{ij} is the Kronecker delta function, p is the pressure, τ_{ij} is the effective viscous stress tensor accounting for the Reynolds stresses, E is the specific total energy, H is the specific total enthalpy, and Q_i is the ith component of the heat flux accounting for turbulent energy transfer, $\vec{Q}$. For turbulent closure, we use the one-equation Spalart–Allmaras negative (SA-neg) turbulence model,

$$\partial_j(\rho v_j \tilde{\nu}) = \partial_j\left(\frac{1}{\sigma}\rho(\nu + \nu')\partial_j \tilde{\nu}\right) - \frac{1}{\sigma}(\nu + \nu')\partial_j \rho \partial_j \tilde{\nu} + \frac{c_{b2}\rho}{\sigma}\partial_j \tilde{\nu}\partial_j \tilde{\nu} + P - D, \tag{4}$$

where ν is the kinematic viscosity, ν' is the part of the turbulence variable viscosity generated by the turbulence variable, $\tilde{\nu}$ is the turbulence variable, σ and c_{b2} are constants, P is a production source term, and D is a destruction source term. We define

$$\nu' = \begin{cases} \nu\chi & \chi \geq 0 \\ \nu\frac{c_{n1}+\chi^3}{c_n 1-\chi^3} & \chi < 0 \end{cases}, \quad \text{with,} \quad \chi = \frac{\tilde{\nu}}{\nu}, \tag{5}$$

where c_{n1} is a constant chosen by the turbulence model. The detailed explanation of each term, including the constants and source terms, can be found in the NASA turbulence modeling resource[1] and the original work of Spalart and Allmaras [19–21].

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 [12,16]. 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,

$$\begin{aligned} u^+(y^+) = \bar{B} + c_1 \log\left(\left(y^+ + a_1\right)^2 + b_1^2\right) - c_2 \log\left(\left(y^+ + a_2\right)^2 + b_2^2\right) \\ - c_3 \operatorname{atan2}\left(b_1, y^+ + a_1\right) - c_4 \operatorname{atan2}\left(b_2, y^+ + a_2\right), \end{aligned} \tag{6}$$

where the coefficients, $\{\bar{B}, c_1, c_2, c_3, c_4, a_1, a_2, b_1, b_2\}$, given are from Allmaras [12] and the plus quantities are given by

$$u^+ = \frac{q_t}{u_\tau} \quad \text{and} \quad y^+ = \frac{\eta u_\tau}{\nu}, \tag{7}$$

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

where u_τ is the friction velocity, q_t 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 [16] to increase accuracy into the wake region of the boundary layer. The ODE is given by

$$\begin{aligned}\frac{\partial}{\partial\eta}\left[(\mu+\mu_t)\frac{\partial q_t}{\partial\eta}\right] &= \left.\frac{\partial p}{\partial\xi}\right|_F + \psi(\eta)\,\rho|_F\left[q_{t,F}\left.\frac{\partial q_t}{\partial\xi}\right|_F + q_{n,F}\left.\frac{\partial q_t}{\partial\eta}\right|_F\right]\\ \frac{\partial}{\partial\eta}\left[\frac{1}{\sigma}(\nu+\tilde\nu)\frac{\partial\tilde\nu}{\partial\eta}\right] &= -\frac{c_{b2}}{\sigma}\left(\frac{\partial\tilde\nu}{\partial\eta}\right)^2 - (P-D),\end{aligned} \tag{8}$$

where ξ is the wall-tangential distance, η is the wall-normal direction, q_t is the tangential velocity, q_n 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. This allows the pressure gradient to be balanced by the viscous terms close to the wall and the inviscid convection terms in the wake region of the boundary layer. This balance between viscous and inviscid terms enables interior forcing points much further into the wake region than the analytical SA wall function. We use the activation function as defined by Berger and Aftosmis [16],

$$\psi(\eta) = \frac{u^+_{\mathrm{SA}}(\eta)}{u^+_{\mathrm{SA}}(F)}, \tag{9}$$

where u^+_{SA} is defined by Eq. (6) and F is the distance from the wall of the forcing point.

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. [17]. The changes to the SA model are as follows:

$$\mu_t = \rho\nu f_{m1}, \tag{10a}$$

$$f_{m1} = \sqrt{\chi^2 + \chi^2_{t0}}, \tag{10b}$$

$$\tilde S = S + \frac{\tilde\nu}{\kappa^2 d^2} f_{m2}, \tag{10c}$$

$$f_{m2} = 1 - \frac{\chi}{1+f_{m1}}, \tag{10d}$$

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} from the original turbulence model. 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,

$$\vec\tau_{w,m} = \rho|\vec v_{\tau,m}|\vec v_{\tau,m}, \tag{11}$$

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

$$\vec v_{\tau,m} = \frac{\vec v_t}{u^+_{\mathrm{WM}}(0)}, \tag{12}$$

$\vec v_t$ is the wall tangential velocity, and u^+_{WM} is analytical wall model velocity, defined by

$$u^+_{\mathrm{WM}}(0) = \frac{\log(\kappa\chi_{t0})}{\kappa} + c, \tag{13}$$

where κ is the von Kármán constant, and c is defined by Ursachi et al. [17]. The wall model also affects the energy equation. Ursachi et al. [17] derived the heat flux of the wall model given the RANS-SA heat flux, $Q_w|_{\mathrm{RANS}}$,

$$Q_w|_{\mathrm{WM}} = Q_w|_{\mathrm{RANS}} - \vec v_t\cdot\vec\tau_{w,m}. \tag{14}$$

Since there is a slip velocity, the heat flux for an adiabatic boundary condition is not zero. Although no heat crosses the boundary, the nonzero heat flux accounts for the heat released by viscous dissipation in the unresolved near-wall region.

3. Methodology

In this section, we describe our discretization and solution strategy. We describe the RANS-SA Newton–Krylov solver and the coupling required for the wall models. We develop a novel implementation of the ODE wall function that solves Eq. (8). Since we solve an ODE for each wall function at each residual evaluation, our method ensures the results are accurate, robust, and computationally inexpensive. The scheme we develop to couple the wall function to the background mesh combines techniques and tricks from the literature. We found that small changes in this coupling scheme have a profound impact on the smoothness and accuracy of results.

Our flow solver, Viscous Aerodynamic Cartesian Cut cells (VACC), is a two-dimensional second-order finite-volume Cartesian cut-cell method [15,22]. The Cartesian meshes are stored as a space-filling curve ordered list of Cartesian cells. 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 centered difference across the adjacent cell states.

In cut cells, the linear state representation is not sufficiently accurate. 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 similarly 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 Fig. 1.

Each boundary face computes a forcing point by drawing a line that starts at the boundary, passing through the cell centroid a distance $h = 1.6\times\max(dx, dy)$ away from the boundary, where dx and dy are the x and y widths of the cell, respectively. The constant, 1.6, was chosen to ensure that the forcing point lies outside the cut cell that contains the no-slip wall boundary condition. The distance h is a constant to compute smooth skin frictions even with varying cut-cell sizes [13]. We refer to the cell containing the forcing point as the 'parent cell'. During a flux evaluation, a state is reconstructed to the forcing point using a stencil that includes the parent cell, the parent cell's nearest neighbors and the parent cell's diagonal neighbors. We fit the forcing point values according to a linear wall tangential approximation and a quadratic wall-normal approximation using Taylor series expansions,

$$\begin{bmatrix} 1/||\vec r_1|| & r_{\xi,1}/||\vec r_1|| & r_{\eta,1}/||\vec r_1|| & 0.5r^2_{\eta,1}/||\vec r_1|| \\ 1/||\vec r_2|| & r_{\xi,2}/||\vec r_2|| & r_{\eta,2}/||\vec r_2|| & 0.5r^2_{\eta,2}/||\vec r_2|| \\ \vdots & \vdots & \vdots & \vdots \\ 1/||\vec r_n|| & r_{\xi,n}/||\vec r_n|| & r_{\eta,n}/||\vec r_n|| & 0.5r^2_{\eta,n}/||\vec r_n|| \end{bmatrix}\begin{bmatrix} Q|_F \\ \partial Q/\partial\xi|_F \\ \partial Q/\partial\eta|_F \\ \partial^2 Q/\partial\eta^2|_F \end{bmatrix} = \begin{bmatrix} Q_1/||\vec r_1|| \\ Q_2/||\vec r_2|| \\ \vdots \\ Q_n/||\vec r_n|| \end{bmatrix}, \tag{15}$$

where $\vec r_i$ is the vector pointing from the forcing point to a cell centroid in its stencil split into wall-normal coordinate components and Q is quan-

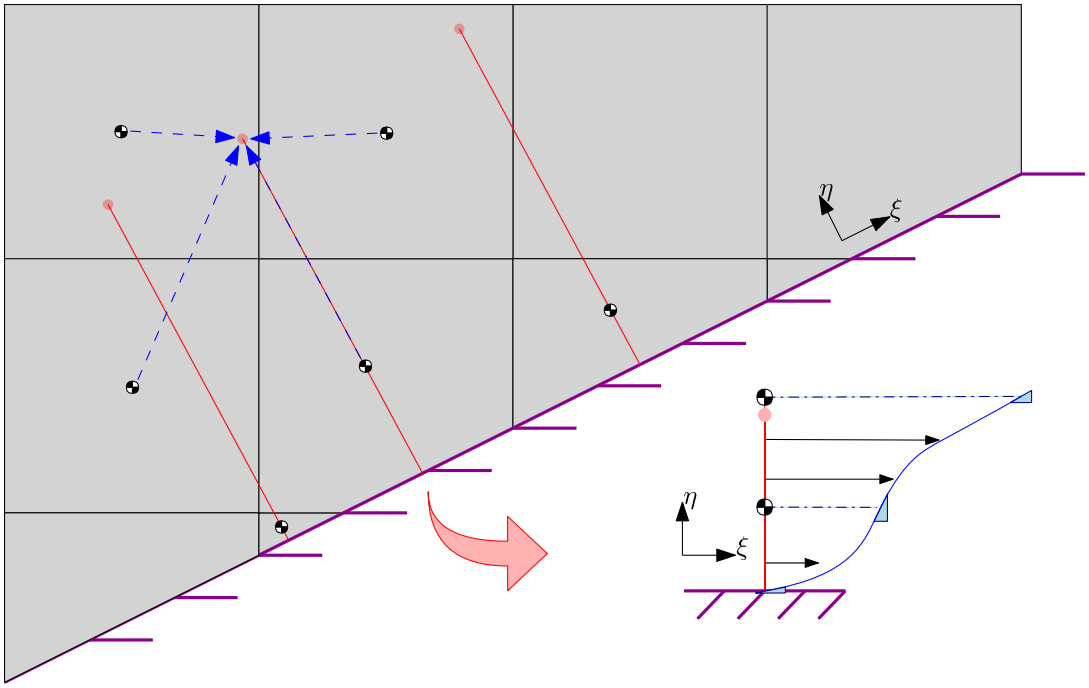


Fig. 1. Coupling between the forcing point and the background mesh. The background mesh provides a state at the forcing point that fits the desired wall function. The wall function then provides states and gradients at various locations for computing viscous fluxes. Not all cells used in this interpolation appear in the figure; additional cells above and to the left complete the stencil.

tity that is needed at the forcing point. The subscripts denote which cell in the forcing point's stencil the state or vector corresponds to. We use η to denote the wall-normal direction and ξ to denote the wall-tangential direction. This convention is shown in Fig. 1. In general, this is an overdetermined system that is solved using least squares. When interpolating states in the boundary layer, changing even a single cell in the interpolation stencil can have a huge impact on the outcome. We weight each equation in the linear system with the inverse distance between the forcing point and a cell centroid to make states closer to the forcing point more influential than those further away. This prevents sudden jumps in the interpolated states when adjacent forcing points shift across parent cell boundaries, changing the interpolation stencil. The SA wall function only needs the wall-tangential velocity interpolated. However, the ODE needs five quantities: the wall-tangential velocity, the density, the pressure gradient, the wall-tangential momentum convection term, and the turbulence variable. Some of these quantities require a gradient. This could be taken from $\partial Q/\partial\xi$ or $\partial Q/\partial\eta$ in our solution to Eq. (15), but to obtain even smoother quantities, we choose to just fit the gradient quantity. In the case of the pressure gradient, we set $Q = \partial p/\partial\xi$. The gradients of Q are not used, as they just provide a basis for the interpolation of the forcing point data.

In most cases, this interpolation procedure works well for the primal quantities like density and tangential velocity. However, cut cells along the wall have notoriously poor gradient stencils and introduce nonuniform discretization errors into the gradient calculation. For gradient quantities like the pressure gradient and convective forces, the accuracy of this interpolation approach captures the noise in the gradient calculation. Fig. 2 shows the tangential velocity and pressure gradient interpolated along a 2D turbulent bump.

This noise can reflect on the skin friction in the solution producing noisy data.

To combat this noise, we use an interpolation scheme for the gradient quantities that is less accurate. We reduce the wall normal reconstruction to linear rather than quadratic and remove the cut cells from the interpolation stencil. Additionally, we remove the inverse distance weighting from the least-squares problem,

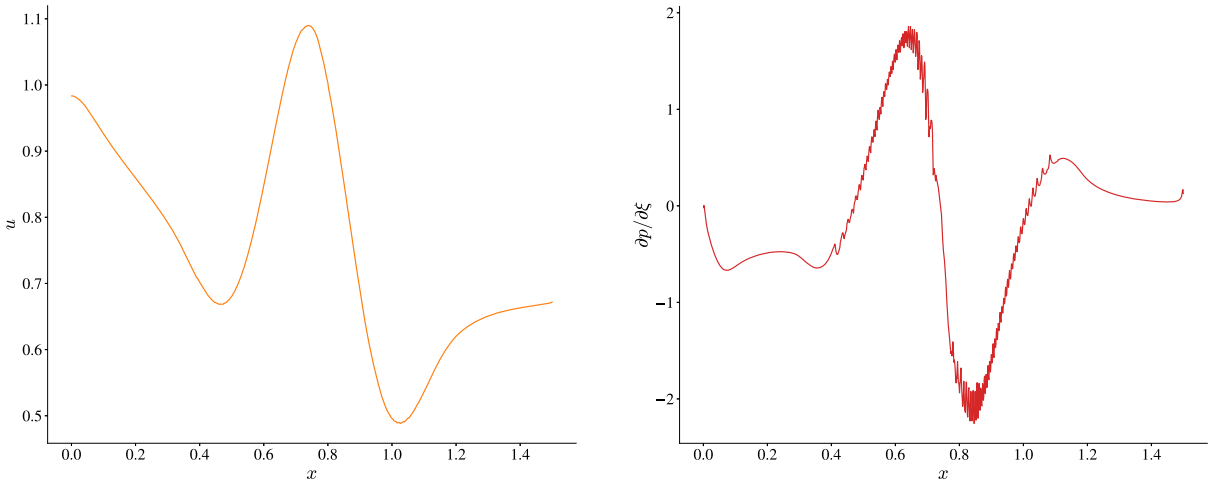
$$\begin{bmatrix} 1 & r_{\xi,1} & r_{\eta,1} \\ 1 & r_{\xi,2} & r_{\eta,2} \\ \vdots & \vdots & \vdots \\ 1 & r_{\xi,n} & r_{\eta,n} \end{bmatrix} \begin{bmatrix} Q|_F \\ \partial Q/\partial\xi|_F \\ \partial Q/\partial\eta|_F \end{bmatrix} = \begin{bmatrix} Q_1 \\ Q_2 \\ \vdots \\ Q_n \end{bmatrix}. \tag{16}$$

Fig. 3 presents the resulting smoothed interpolated pressure gradient.

Once the wall function is defined, it can be used to inform the fluxes for the background cut cell. The wall function is coupled back into the global flow solution in three ways: to evaluate the *shear stress at the wall*, to compute the vorticity for the *SA source term* in cut cells and adjacent cells, and to evaluate the velocity gradients for *cut-cell viscous fluxes*. The wall shear is computed straight from the wall function by evaluating its velocity gradient, $\partial q_t/\partial\eta$, at $\eta = 0$ and then rotated back into the global coordinate frame. The vorticity in cut cells and cells adjacent to cut cells is set to the velocity gradient at the cell centroid's distance from the no slip face that holds the wall function. We assume that the contribution of the normal velocity gradient, $\partial q_n/\partial\xi$, is zero.

The viscous flux evaluation requires additional care in and around cut cells. As described 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 cell centroids are not 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. Fig. 4 shows this process.

In the interior cells, this recentering is done by projecting the state using the least-squares gradient. For the cut cells, this projection is identical for the density, pressure and turbulence variable. The velocity and velocity gradient at face centroids is obtained by evaluating the wall function a distance from the no slip wall that matches the face centroid's distance from the wall. The wall functions are used to inform viscous fluxes on every face adjacent to a cut cell. For cut faces adja-



(a) u (b) $\partial p/\partial \xi$

Fig. 2. Interpolated tangential velocity and tangential pressure gradient around a two-dimensional bump. Oscillations are clearly visible in the pressure gradient not visible in the tangential velocity. These oscillations can cause oscillations in the final skin friction.

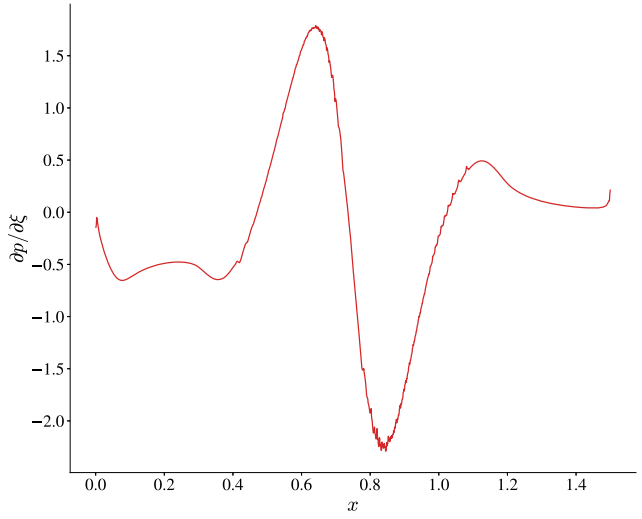


Fig. 3. Smoothed pressure gradient interpolation around a two-dimensional bump.

cent to two cut cells, the two associated wall function evaluations are averaged.

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,

$$\frac{\partial \mathbf{u}}{\partial t} + \vec{\mathbf{F}}(\mathbf{u}) = \mathbf{S}(\mathbf{u}), \tag{17}$$

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

$$\mathbf{R}_{\text{cell}} = \sum_{e=1}^{N_e} \hat{\mathbf{F}}\Big|_e l_e - \mathbf{S}_{\text{cell}} A_{\text{cell}}, \tag{18}$$

where e indicates edges of the cell and $\hat{\mathbf{F}}$ is a numerical flux computed at an edge, e. A_{cell} and S_{cell} are the area and source terms evaluated for a given cell. The Newton method solves linear systems at each step,

$$\left.\frac{\partial \mathbf{R}}{\partial \mathbf{u}}\right|_{\mathbf{u}^n} \Delta\mathbf{u} = -\mathbf{R}(\mathbf{u}^n), \tag{19}$$

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 [23–25]. We use a backward-Euler time step,

$$\frac{\mathbf{A}}{\Delta \mathbf{t}}\left(\mathbf{u}^{n+1} - \mathbf{u}^n\right) + \mathbf{R}\left(\mathbf{u}^{n+1}\right) = \mathbf{0}, \tag{20}$$

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 [22]. 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, and 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-stable and therefore will always be linearly stable as long as the system has 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 helps nonlinear convergence. The nonlinear preconditioner is,

$$\mathbf{P}_{t,\text{local}} = \frac{\partial \mathbf{u}_{\text{conservative}}}{\partial \mathbf{u}_{\text{primitive}}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ u & \rho & 0 & 0 & 0 \\ v & 0 & \rho & 0 & 0 \\ (u^2+v^2)/2 & \rho u & \rho v & 1/(\gamma-1) & 0 \\ \tilde{\nu} & 0 & 0 & 0 & \rho \end{bmatrix}, \tag{21}$$

where γ is the specific heat ratio, u is the x-velocity, and v is the y-velocity. We premultiply the time step term in Eq. (20) by a block diagonal $\mathbf{P}_t$ matrix made up of matrices from Eq. (21).

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

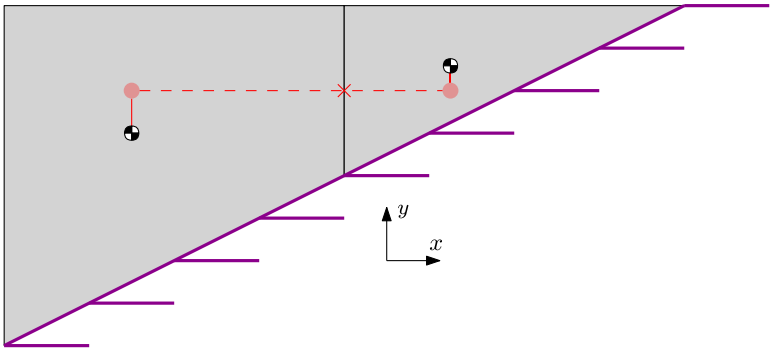


Fig. 4. The recentering process used to compute gradients at face centroids not aligned with the cell centroids.

Eq. (4) by the same value,

$$\begin{aligned}&\partial_t(\rho\tilde{\nu}_{\text{norm}}) + \partial_j(\rho u_j \tilde{\nu}_{\text{norm}}) \\ &\quad = \partial_j\left(\frac{1}{\sigma}\rho(\nu+\nu')\partial_j\tilde{\nu}_{\text{norm}}\right) - \frac{1}{\sigma}(\nu+\nu')\partial_j\rho\partial_j\tilde{\nu}_{\text{norm}} \\ &\quad + \sqrt{\tilde{\nu}_\infty}\frac{c_{b2}\rho}{\sigma}\partial_j\tilde{\nu}_{\text{norm}}\partial_j\tilde{\nu}_{\text{norm}} + \frac{P-D}{\sqrt{\tilde{\nu}_\infty}}.\end{aligned} \tag{22}$$

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

To converge to steady state, we linearize Eq. (20) about $\mathbf{u}^n$ with the preconditioner from Eq. (21), which yields

$$\left(\frac{\mathbf{A}}{\Delta\mathbf{t}}\mathbf{P}_t + \left.\frac{\partial\mathbf{R}}{\partial\mathbf{u}}\right|_{\mathbf{u}^n}\right)\Delta\mathbf{u} = -\mathbf{R}(\mathbf{u}^n). \tag{23}$$

As the simulation proceeds, the time step increases towards infinity and Eq. (23) recovers the full Newton step in Eq. (19).

Each nonlinear step requires solving the linear system in Eq. (23). To avoid forming the full residual Jacobian matrix, we use the generalized minimum residual method (GMRES) [26]. 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 $\mathbf{u}^n$,

$$\left.\frac{\partial\mathbf{R}}{\partial\mathbf{u}}\right|_{\mathbf{u}^n}\mathbf{v} \approx \frac{\mathbf{R}(\mathbf{u}^n + \epsilon\mathbf{v}) - \mathbf{R}(\mathbf{u}^n)}{\epsilon}. \tag{24}$$

We define ϵ based on the work in Brown and Saad [27] and Yildirim et al. [25],

$$\epsilon = \begin{cases} e_{\text{rel}}\mathbf{v}\cdot\mathbf{u}^n/||\mathbf{v}||_2^2 & \text{if} \quad |\mathbf{v}\cdot\mathbf{u}^n| > e_{\text{min}}||\mathbf{v}||_1 \\ e_{\text{rel}}e_{\text{min}}\text{sign}(\mathbf{v}\cdot\mathbf{u}^n)||\mathbf{v}||_1/||\mathbf{v}||_2^2 & \text{otherwise,}\end{cases} \tag{25}$$

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

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 use a block-diagonal point-Jacobi preconditioner, where each block corresponds to one cell. All the diagonal blocks of the residual Jacobian are computed analytically and 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 Newton–Krylov solver needs a globalization strategy to converge reliably. Newton methods converge residuals quadratically near a solution but can stall when seeking a point close enough for quadratic convergence. We use pseudo-transient continuation and a physicality line search to globalize our Newton–Krylov method [23,25]. The line search ensures the solution remains physical, since Newton–Krylov is not a time integration scheme and updates may produce nonphysical solutions. Pseudo-transient continuation increases the time-step in Eq. (23). As this time-step grows, the linear system converges to the exact Newton update in Eq. (19). This uses physical evolution to drive the state towards the quadratic convergence region before the full Newton step accelerates convergence.

3.2. SA wall function

The SA wall function is a traditional wall function. The friction velocity, u_τ, in Eq. (6) is fit to the forcing point data using a Newton method. Once u_τ is computed, Eq. (6) 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 present a novel approach to computing a solution to this ODE that is robust, accurate, and computationally inexpensive. We implemented a high-order continuous Galerkin (CG) method with Mesh Optimization via Error Sampling and Synthesis (MOESS) for solving Eq. (8). The convection terms in Eq. (8) are constants with respect to the state variables, giving an ODE with no advection terms. This lends itself to a CG formulation. The solver uses Newton–Krylov with pseudo-transient continuation similar to the background flow solver 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 Eq. (17) we can rewrite Eq. (8) as

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

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

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 returns a step that causes one of the turbulence variable states to become negative, the CFL number is reduced and the linear step is rejected. If the CFL number 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. A constant, β, is used to reduce the magnitude of the ξ-momentum source term in Eq. (8) as follows:

$$\frac{\partial}{\partial\eta}\left[(\mu+\mu_t)\frac{\partial u}{\partial\eta}\right] = \beta\left(\left.\frac{\partial p}{\partial\xi}\right|_F + \psi(\eta)\,\rho|_F\left[u_F\left.\frac{\partial u}{\partial\xi}\right|_F + v_F\left.\frac{\partial u}{\partial\eta}\right|_F\right]\right), \tag{28}$$

where $\beta \in [0, 1]$. When an ODE solution fails, β is reduced by 0.01. The momentum balance between the pressure and density is crucial for this wall function, and therefore we only apply β across the entire source term. 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.

To ensure accurate solutions for a wide variety of boundary conditions, we implemented MOESS in conjunction with our CG solver [28]. 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 obtain an accurate wall shear stress.

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

$$\delta J \approx -\boldsymbol{\Psi}_h^T \mathbf{R}_h(\mathbf{U}_h^H), \tag{29}$$

where δJ is the error of some functional J with respect to each state, $\boldsymbol{\Psi}_h$ is the fine-space adjoint, and $\mathbf{R}_h(\mathbf{U}_h^H)$ 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 dimensions, 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

$$\delta l = \sqrt{\delta\vec{x}^T \mathcal{M} \delta\vec{x}}, \tag{30}$$

and in 1D,

$$\delta l = \delta x \sqrt{\mathcal{M}}. \tag{31}$$

In an ideal metric-conforming mesh in 1D, every element should have unit length.

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. This metric field is stored at the previous mesh nodes. We assume the logarithm of the metric varies linearly between two nodes [29],

$$M_x = M_a\left(\frac{M_b}{M_a}\right)^{\frac{x-a}{b-a}}, \tag{32}$$

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

$$L_e = \begin{cases} \frac{L_a - L_b}{\log L_a/L_b} & |L_a - L_b| > 0.001 \\ \frac{L_a + L_b}{2} & \text{otherwise} \end{cases}, \quad \text{where } L_a = l_e\sqrt{M_a}, L_b = l_e\sqrt{M_b}. \tag{33}$$

For 1D CG, the total number of degrees of freedom 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 of unit length under the metric. This is not always the case, so we use Eq. (33) to compute the piecewise integral of the metric over the entire domain. This is divided by the number of elements requested to obtain the constant-valued metric length for all the elements. To generate the mesh, we start at $\eta = 0$ and place points at the constant-valued metric length distances apart until the other end of the domain is reached.

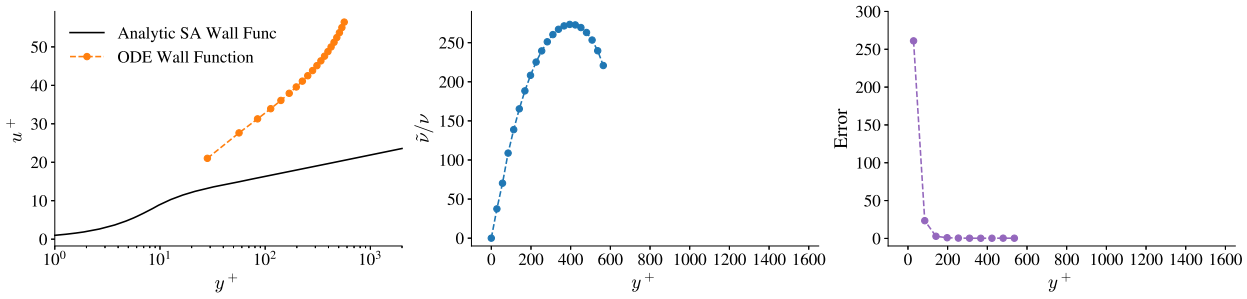
Eq. (33), together with a Newton solver is used to compute the location of each point in the new mesh. Each point in the new mesh will lie between two points on the background mesh, $[a, b]$. To place that point, the interval $[a, b]$ is determined by checking the integral of the metric from the previously 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, Eq. (33) is cast as a residual, and a Newton solver identifies η such that the metric length between the previous new mesh point and this point is the desired length.

Fig. 5 shows the progression of MOESS on a $p = 2$ solution of the ODE solution starting with a uniformly spaced mesh.

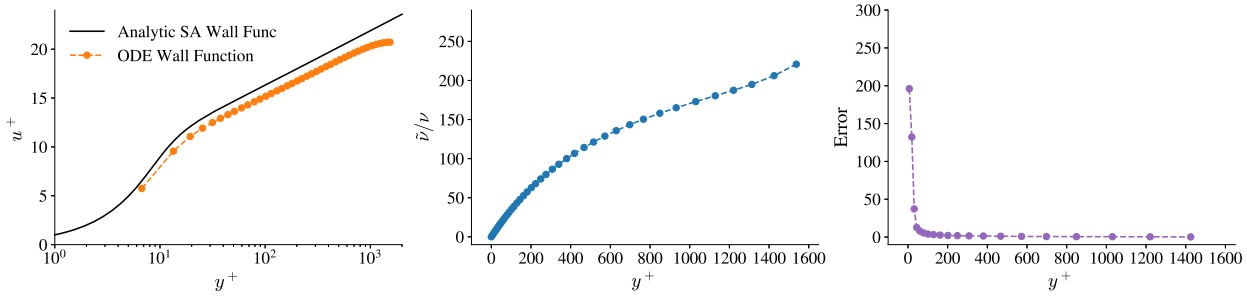
The boundary conditions for this case come from the 2D turbulent bump case on the NASA turbulence modeling resource [19]. The finest CFL3D solution was used to obtain forcing point data downstream of the bump. Each MOESS cycle consists of a flow solution, followed by the MOESS mesh optimization. Fig. 5(a) indicates that the naive uniform mesh yields very poor results. The velocity profile does not come close to matching what we expect. Even with this mesh, the Newton–Krylov solver is robust enough to converge the system by dropping the momentum source terms. After the first adaptation, the number of cells is doubled, and the mesh points start sliding towards the wall. This is enough to allow the momentum source terms to be turned back on. The velocity profile converges to what we would expect. The forcing point approaches the wake region, and the ODE is able to begin predicting this transition. Although the physical distance of the forcing point never changes throughout these optimizations, the ODE wall function computes the friction velocity using its own solution. The first solution has a final y^+ just under half of the others because it is not able to obtain an accurate friction velocity.

4. Results

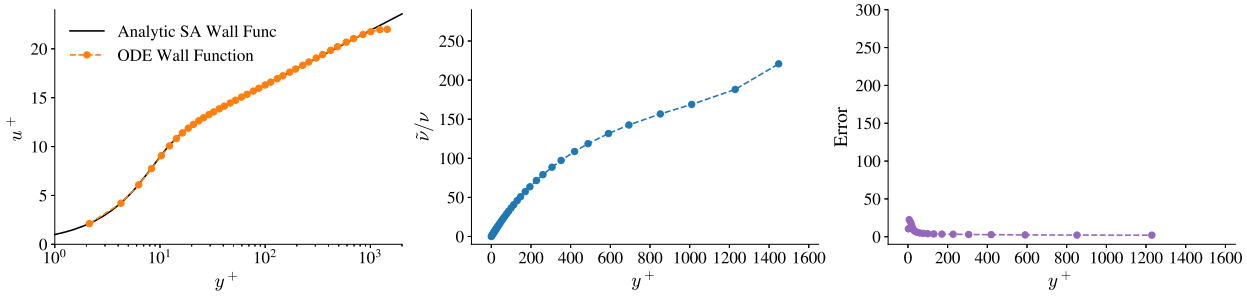
To verify our methodology, we work through an increasingly complex series of verification test cases from the NASA turbulence modeling resource (TMR) [19]. The first case is a grid-aligned flat plate with no pressure gradient. After verifying our method on a grid with no cut cells, we rotate the flat plate 15° into the mesh domain. This allows us to solve the same zero pressure gradient case with the addition of



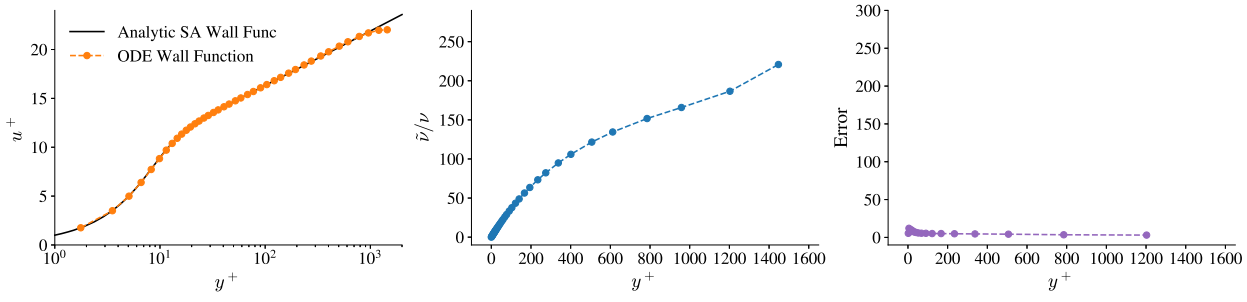
(a) Initial mesh with 10 linearly spaced $p = 2$ elements.



(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.

Fig. 5. 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 Eq. (8). It is not necessarily accurate given the forcing point location.

cut cells, testing the strength of our viscous stencil. Next, we verify the turbulent bump case, which introduces a pressure gradient. This allows the ODE wall model to demonstrate the benefits of including a pressure-momentum balance source term in the wall function. Finally, we test a NACA 0012 airfoil to demonstrate the method on a realistic external aerodynamic case.

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

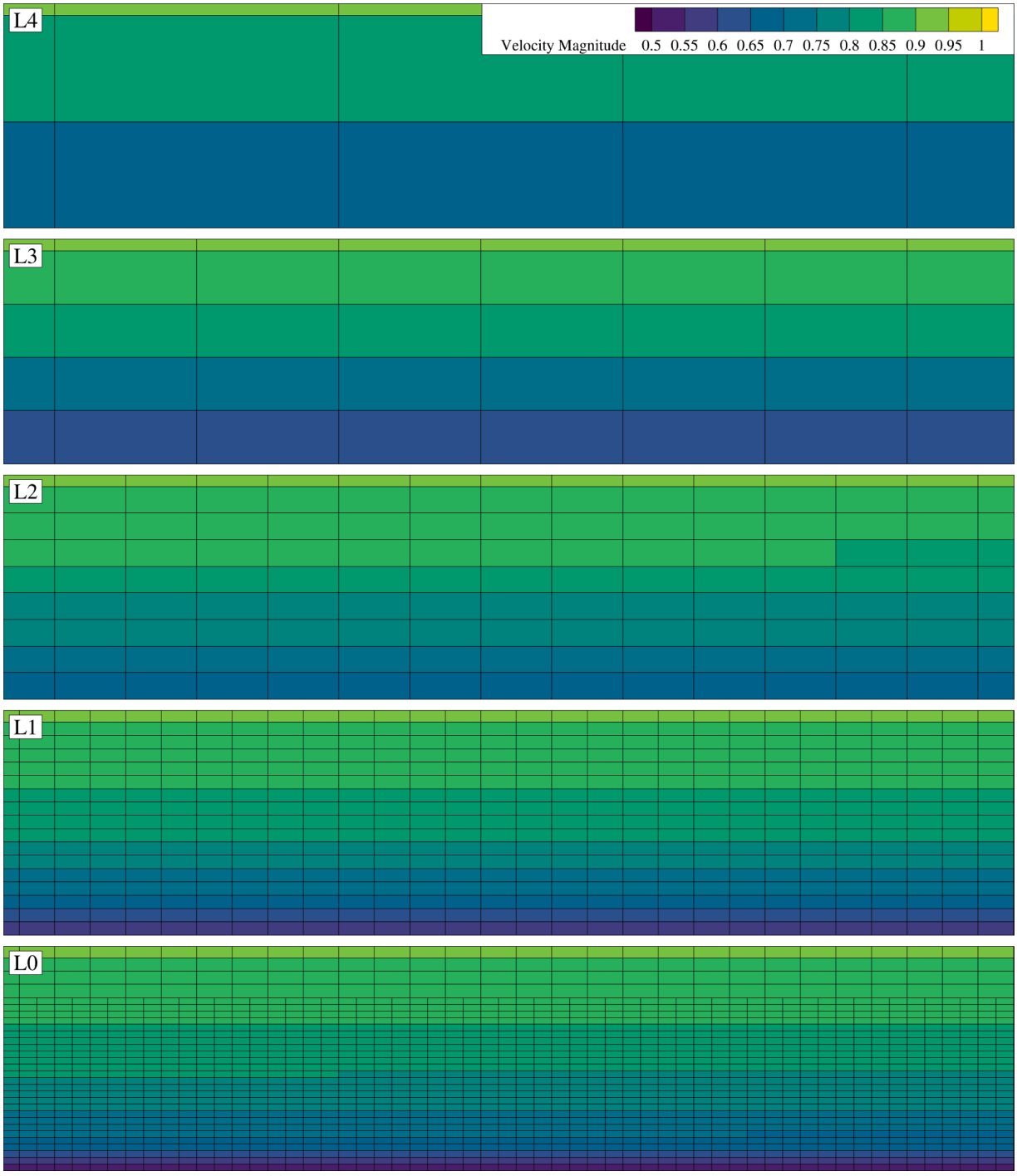


Fig. 6. Comparison of the mesh resolutions for the mesh families. The left side of this figure corresponds to $Re_x = 10^7$. The off-wall spacing of the first cell in the L0 mesh is 4×10^{-4}. The plate is a length of one with a Reynolds number of 10^7. As the mesh is refined, the boundary layer becomes clearer.

run a sequence of meshes for this case, where the L0 mesh is the finest and L4 is the coarsest. Fig. 6 shows the full mesh family with velocity magnitude contours.

We only present detailed results for the finest three meshes: L0, L1, and L2. The plate is length one. The L0 mesh has an initial off-wall spacing of $y \approx 4 \times 10^{-4}$. On each subsequent mesh, isotropic coarsening was performed to double the initial off-wall spacing. The far-field retains the same mesh size, so each level is not an exact factor of 4 decrease in the number of cells. The L0 mesh has approximately 320k cells. The L2 mesh has approximately 84k cells. Fig. 7 presents the forcing point y^+ values for these mesh sizes.

Fig. 8 presents velocity profiles for all three methods at one stations.

The two wall functions match the expected profile well. In both cases, there is a single point that is not on the profile, which 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 converges to the appropriate solution as the mesh is refined. WMRANS needs additional resolution than the wall function approaches, even though the profile it creates is more benign. None of the above approaches exhibit any viscous overshoot.

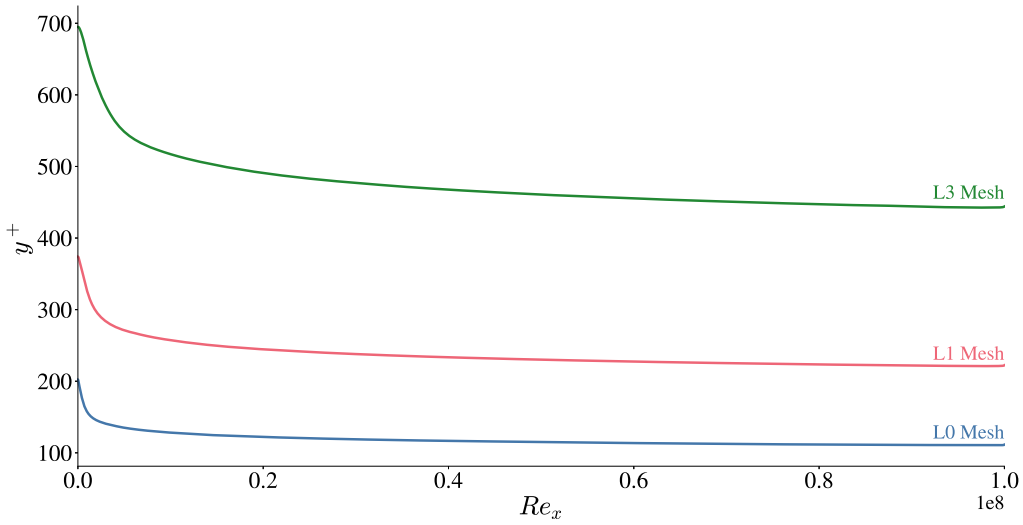
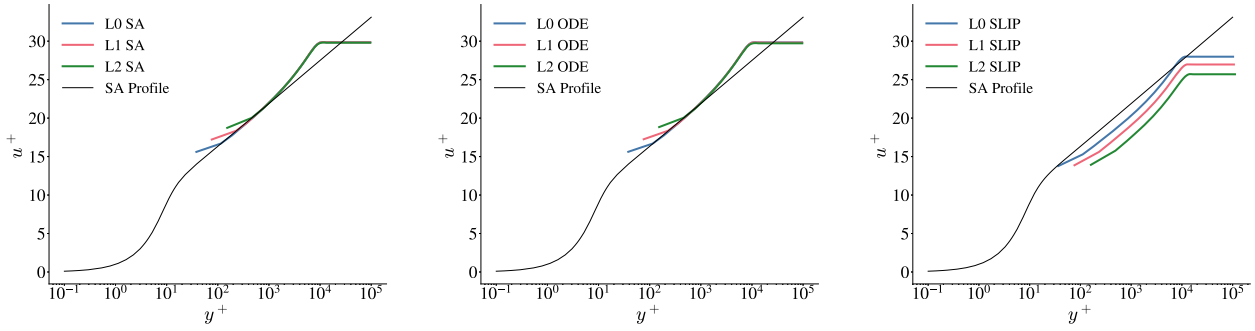
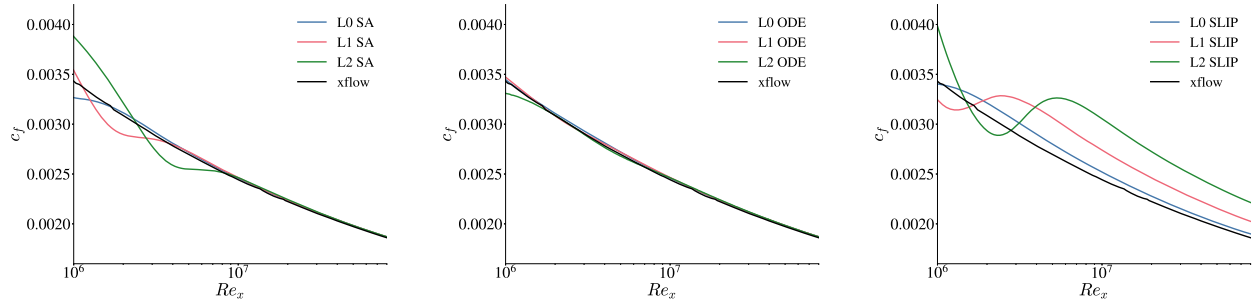


Fig. 7. Forcing point y^+ values on the ODE wall function solution for the finite flat plate case. The L0 mesh has an initial off-wall spacing of $y \approx 4 \times 10^{-4}$. The plate is length one with a Reynolds number of 10^7. Each subsequent level of mesh doubles the off-wall spacing. The $y+$ values at the forcing point indicate that they are well into the wake region on the coarse mesh.



(a) SA wall function velocity profiles (b) ODE wall function velocity profiles (c) WMRANS velocity profiles

Fig. 8. Velocity profiles for the zero pressure gradient finite flat plate case taken at $Re_x = 2 \times 10^7$. The plate is ten reference lengths long. The SA and ODE wall functions match the expected velocity profile. The WMRANS profiles converge to the expected result as the mesh is refined.

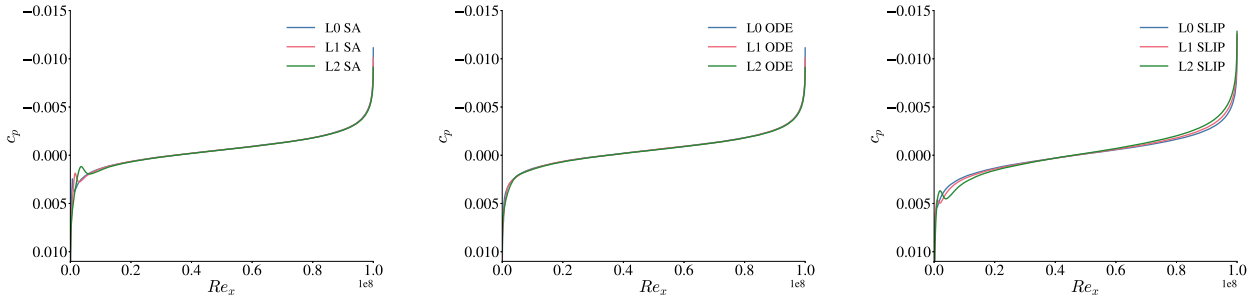


(a) SA wall function skin friction (b) ODE wall function skin friction (c) WMRANS skin friction

Fig. 9. Skin friction coefficient for the finite flat plate case plotted against xflow, an output-adaptive high-order discontinuous Galerkin flow solver [30]. The SA wall function exhibits clear mesh convergence, with skin friction start-up oscillation moving towards the front of the plate as the mesh is refined. The ODE wall function does not exhibit the same start-up oscillations because it can accept forcing points further into the boundary layer. The WMRANS struggles to reach the correct asymptotic skin friction on the same level mesh as the two wall functions.

Figs. 9 and 10 show the skin frictions and pressure coefficients, respectively.

We use xflow, an output-adaptive high-order discontinuous Galerkin flow solver, as the truth value [30]. The biggest difference 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 that as the mesh is refined, the skin friction converges to the asymptotic value sooner. The ODE appears to avoid start up behavior similar to WMRANS and the SA wall function because it can accept forcing points in the wake region. This allows the wall function to accurately predict the skin friction with much coarser meshes. Again,



(a) SA wall function surface pressure (b) ODE wall function surface pressure (c) WMRANS surface pressure

Fig. 10. Pressure coefficients for the finite flat plate case. We see the same mesh converging startup oscillations as in the skin friction for both the WMRANS and the SA wall function. The ODE wall function does not exhibit startup oscillations as it can accept forcing points much further from the wall.

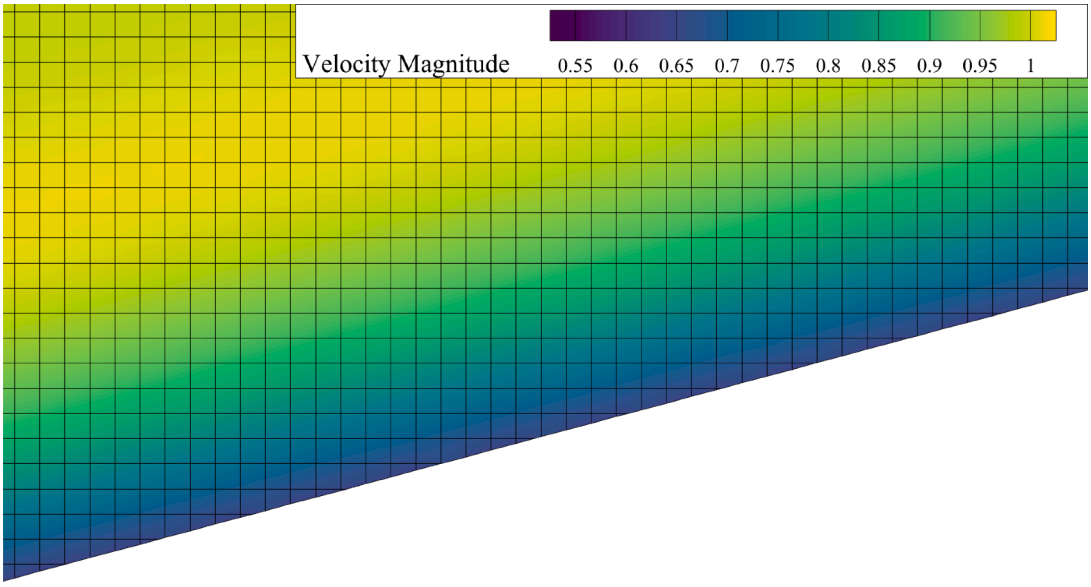


Fig. 11. L1 mesh for the rotated flat plate case. This image is taken from $Re_x \approx 10^7$. The rotated coordinate system introduces cut cells.

we see the same behavior with the WMRANS approaching the expected solution as the mesh resolution increases.

4.2. Rotated finite flat plate

This is the same case as the grid-aligned finite flat plate, but with cut-cells (Fig. 11).

Rather than placing the flat plate along the bottom boundary of the domain, it is set into the domain at a 15° angle. The L0 mesh for this case is one level finer than the grid aligned case. The mesh sequence gets coarser as the mesh number increases with L3 being the coarsest mesh. We used the same conditions as the wall-aligned flat plate (Mach 0.2 and Reynolds number 10^7).

Fig. 12 shows the velocity profiles for the rotated finite flat plate.

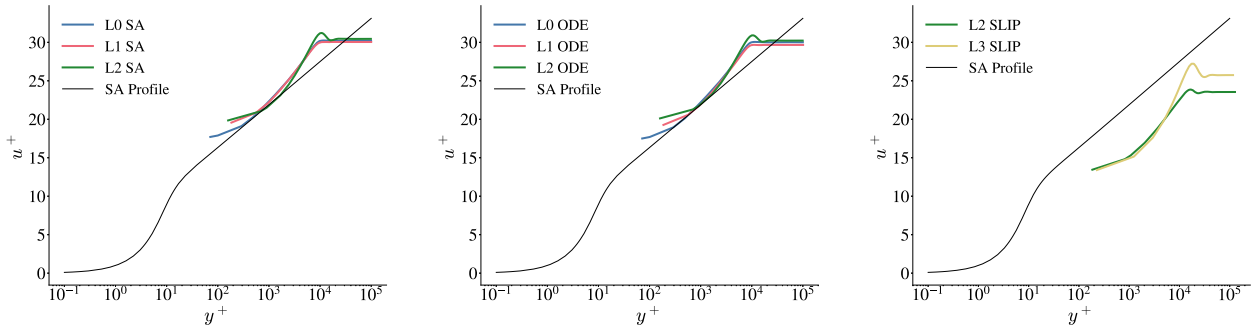
The solver struggles considerably more when cut cells are introduced. The gradient stencils in these cells are degraded and the principal directions are no longer aligned with the mesh. We still see similar behavior in the convergence of the results: as the mesh is refined, all solutions approach the expected results.

Fig. 13 shows the skin-friction coefficients for the rotated flat plate case.

The skin frictions are more noisy for the ODE wall function and considerably more noisy for the WMRANS. 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. On a case with no pressure gradient, such as this one, these oscillations appear more pronounced than on case with an existing pressure gradient. This behavior is independent of the method we choose to resolve the wall, even with the analytical SA function.

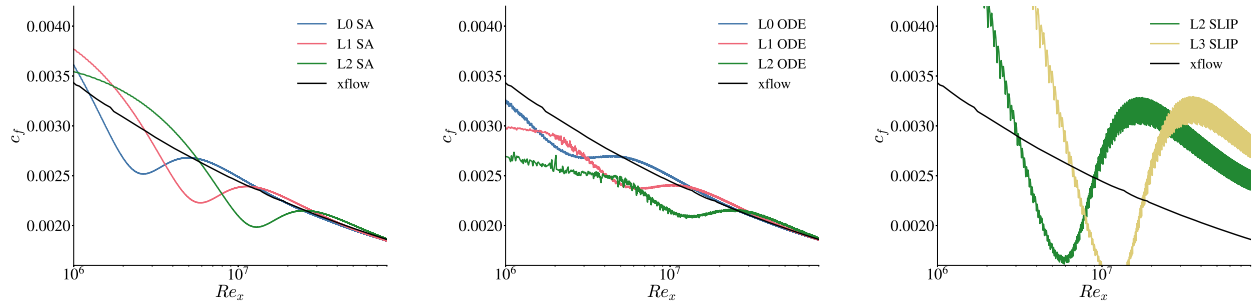
The SA wall function is able to hide the oscillations in the skin friction because the only information it uses at the forcing point is the tangential velocity momentum, which does not oscillate. The ODE wall function contains $\partial p/\partial \xi$ in the ξ-momentum source term and is directly affected 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 converges to a smoother result along the plate. The slip wall struggles with these oscillations, particularly in converging solutions on the finer grids. As seen in the grid-aligned flat plate, the WMRANS needs more resolution than the wall functions. The WMRANS suffers from rapidly changing sizes of cut cells along the wall, but does not have a method similar to the wall functions to smooth out the noisy states. It appears that the WMRANS requires more care in the construction of viscous terms in the cut-cell stencils. The WMRANS susceptibility to both the irregular cut-cell stencils and the need for more wall normal resolution appear to make it a poor choice when used in isolation for high Reynolds number RANS flows on cut-cell meshes. Tamaki et al. showed that a similar WMRANS



(a) SA wall function velocity profiles (b) ODE wall function velocity profiles (c) WMRANS velocity profiles

Fig. 12. Velocity profiles for the zero pressure gradient finite flat-plate case with cut cells. The plate is ten reference lengths long. This case was run at Mach 0.2 and Reynolds number 10^7. The SA and ODE wall function approaches match the expected solution. The WMRANS demonstrates numerical difficulties with the cut-cell meshes. There is no longer obvious mesh convergence. For each method, the coarser meshes demonstrate viscous overshoot.



(a) SA wall function skin friction (b) ODE wall function skin friction (c) WMRANS skin friction

Fig. 13. The skin friction for the rotated finite flat plate case plotted against xflow, a high-order discontinuous Galerkin flow solver [30]. The SA wall function displays almost identical skin friction profiles to the grid aligned flat plate. The ODE wall function displays the same trends, but introduces small oscillations into the skin friction due to oscillation in the background pressure and momentum gradients. The WMRANS exhibits stronger oscillations than the ODE, but lacks the ability to converge to the correct asymptotic skin friction for the same mesh level.

approach can be used in conjunction with a wall function on cut-cell meshes [18].

4.3. 2D bump

We ran the 2D bump-in-channel verification case from the NASA turbulence modeling resource [19], at Mach 0.2 and Reynolds number 3×10^6. This case introduces a pressure gradient into the solution. Again, the L0 mesh is the finest and the L2 mesh is the coarsest. The L0 mesh has approximately 220k isotropic cells, and the L2 mesh has approximately 59k isotropic cells. Fig. 14 shows the y^+ values of the wall-function forcing points along the surface of the bump.

It is clear, particularly on the coarse mesh, that the forcing point is entering the wake region of the boundary layer. Fig. 15 presents a comparison of the eddy viscosity generated by the bump.

The eddy viscosity profiles from VACC are computed on the L1 mesh with the SA wall function. VACC exhibits good agreement with CFL3D.

Fig. 16 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.

Fig. 17 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 very small spurious pressure spikes around triangle cut cells that can barely be seen at this plotting resolution. This is a known phenomena that exists for Cartesian cut-cell solutions on inviscid solutions.

Fig. 18 presents the skin frictions along the bump.

The SA analytical wall function matches the CFL3D solution well. The phase shift seen in the skin frictions occurs because the SA wall function assumes no pressure gradient. As the mesh resolution increases, the skin friction approaches the expected profile. The ODE wall function follows the general trend but is noisy. As the mesh is refined the magnitude of the noise is reduced. The ODE allows the coarser meshes to fix the phase shift without needing additional mesh resolution.

The downturn in the ODE skin friction near the front of the no-slip wall is due to the localized nature of the wall functions. Near the front of the plate, the boundary layer is extremely small and the forcing points lie well into the inviscid region of the flow. This makes all of the forcing point data look identical even though the boundary layer thickness is not. Since the wall functions do not communicate with those in front of or behind them in the wall tangential direction, they have no notion of how thick the boundary layer should be when given data that lies outside the boundary layer. The SA and ODE wall functions have the same issue. However, the ODE under-predicts the skin friction, and the SA wall model over-predicts the skin friction. In both cases, the skin friction at the leading edge of the plate converges under mesh refinement.

4.4. NACA 0012

The final verification case we ran was the 2D NACA 0012 from the NASA turbulence modeling resource [19]. It is run at Mach 0.15 and Reynolds number 6×10^6 at 10° angle of attack. This case introduces a stagnation point more typical of external, aerodynamic, flows. It also introduces a much wider range of $y+$ values with points on the pressure side much further outside the boundary layer than those on the suction side. The L0 mesh is the finest with about 430k cells. Fig. 19 shows the L1 mesh, which has about 258k cells.

The L2 mesh is the coarsest with around 126k cells.

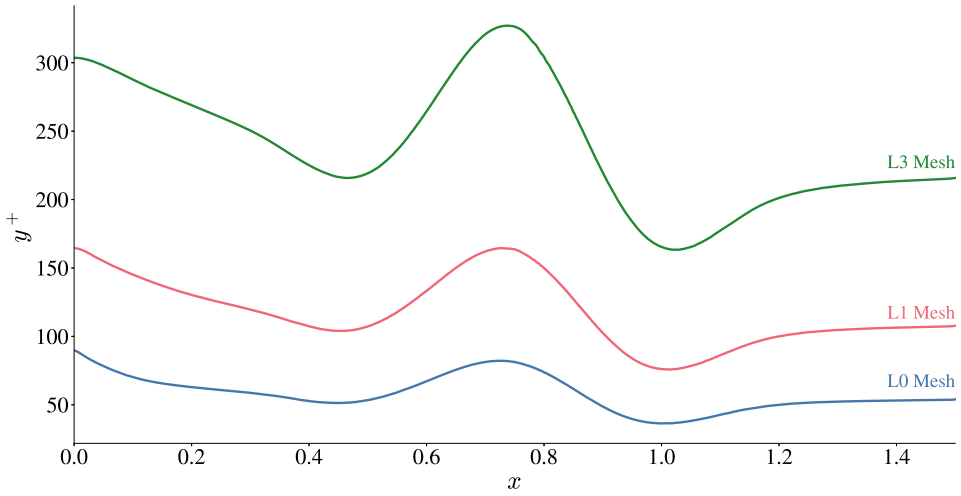


Fig. 14. Forcing point y^+ values on the ODE wall function solution for the turbulent bump case. The forcing points are well into the wake region of the boundary layer, particularly on the coarse mesh.

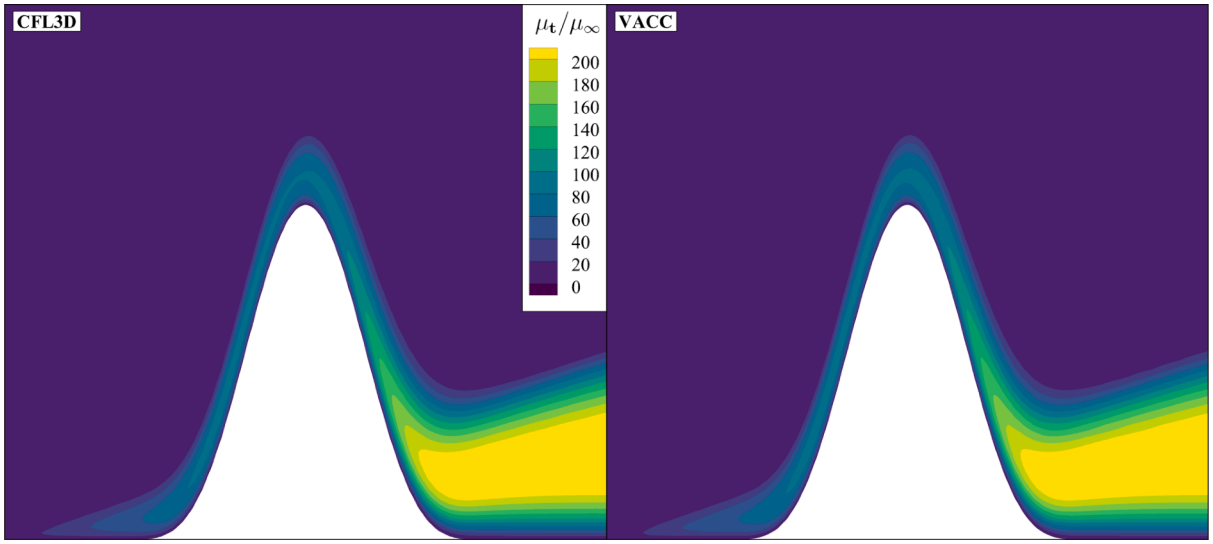


Fig. 15. Turbulent eddy viscosity contours around the bump. The conditions for this case come from the NASA turbulence modeling resource. It was run at Mach 0.2 with a Reynolds number of 3×10^6. The CFL3D solution matches the SA wall function on the L1 mesh. The y-axis is scaled to be approximately twenty times larger than the x-axis.

Fig. 20 presents the pressure coefficients for the SA and ODE wall functions on the NACA 0012 airfoil.

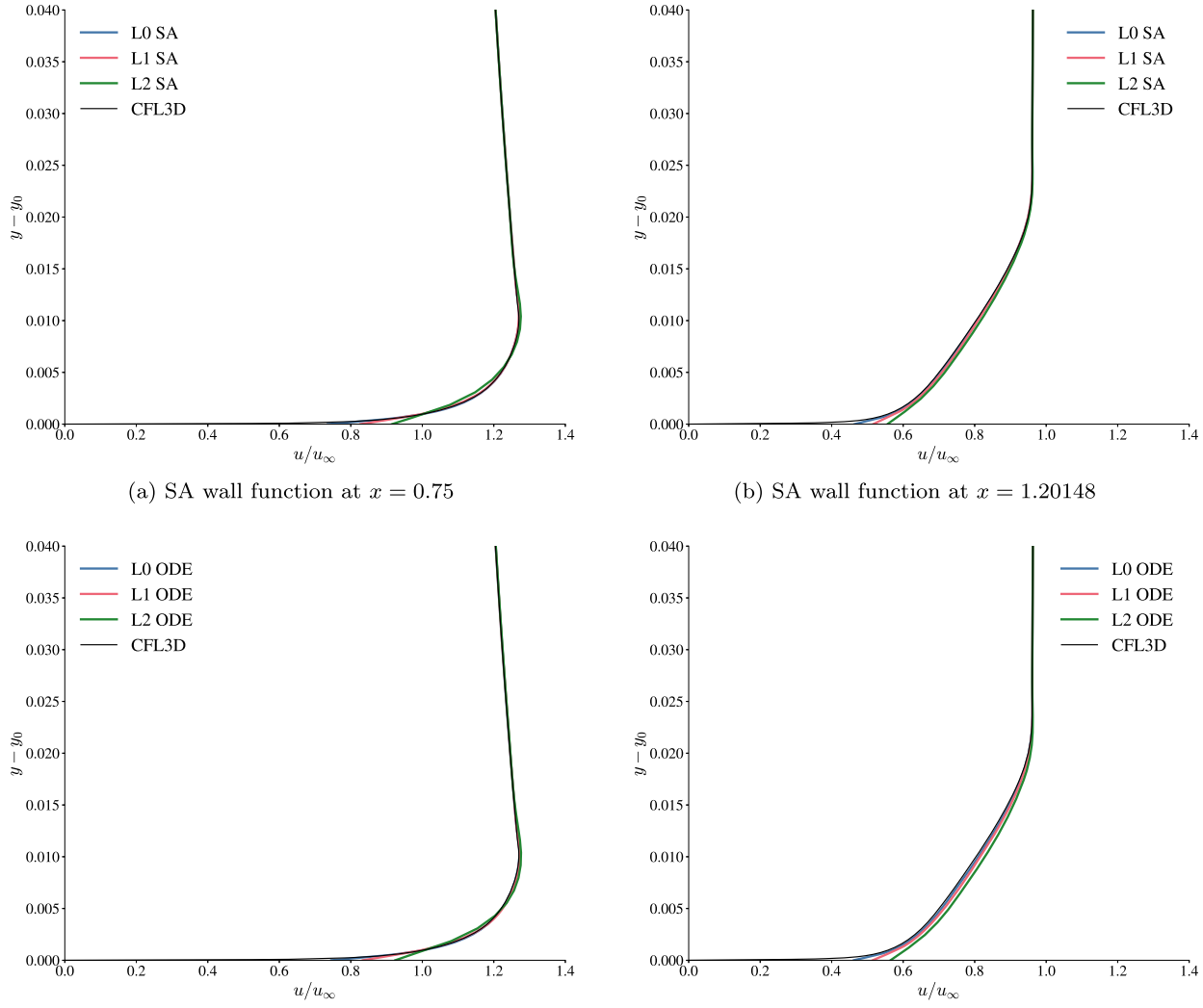
The pressure coefficients match the CFL3D solutions for both wall models at all mesh resolutions. Fig. 21 presents the skin friction coefficients on the NACA 0012 airfoil.

These skin friction plots are where we begin to see the immense benefits of the momentum balance source term in the ODE wall function. The SA skin frictions on the upper surface struggle to converge to the expected skin friction values. The L2 mesh skin friction is not able to recover from the suction peak until around 50 % of the chord. Even on the finest mesh, the skin friction does not recover until around 15 % of the chord. On the other hand, the ODE wall function is able to match the upper surface skin friction before 15 % of the chord on even the coarsest mesh. Both wall functions exhibit mesh convergence, trending towards the CFL3D solution as the mesh is refined.

Skin friction oscillations are smaller for this case when compared to the turbulent bump and the rotated flat plate. As the strength of the global pressure gradient across the geometry increases, the noise visible in the local pressure gradient is reduced. The nonuniform discretization error in the cut cells remains of constant magnitude. When added to stronger pressure gradients, its effect shrinks.

Fig. 22 presents x-velocities and the eddy viscosities immediately aft of the trailing edge of the airfoil.

The tangential velocities match the CFL3D solution well. Both the SA and ODE wall functions converge towards the CFL3D solution as the mesh is refined. The wall functions were not used to extract these lines, the velocity profiles come from the background mesh. The SA wall function eddy viscosity matches the CFL3D eddy viscosity well on the finest mesh. The ODE wall function is better able to capture the eddy viscosity going to zero at $y = 0$ than the SA wall function on coarser meshes. However, it fails to reach the peak on the upper surface. We have found that this behavior is dependent upon turbulence variable profile startup strategies near the leading edge. Choosing different coupling methods can have a profound impact on the startup of the boundary layer where the wall function forcing points are too far outside of the boundary layer to have correct physics. This is true of both the SA and ODE wall functions, but because the ODE wall function depends on more states in the background mesh, it often displays these differences more readily.



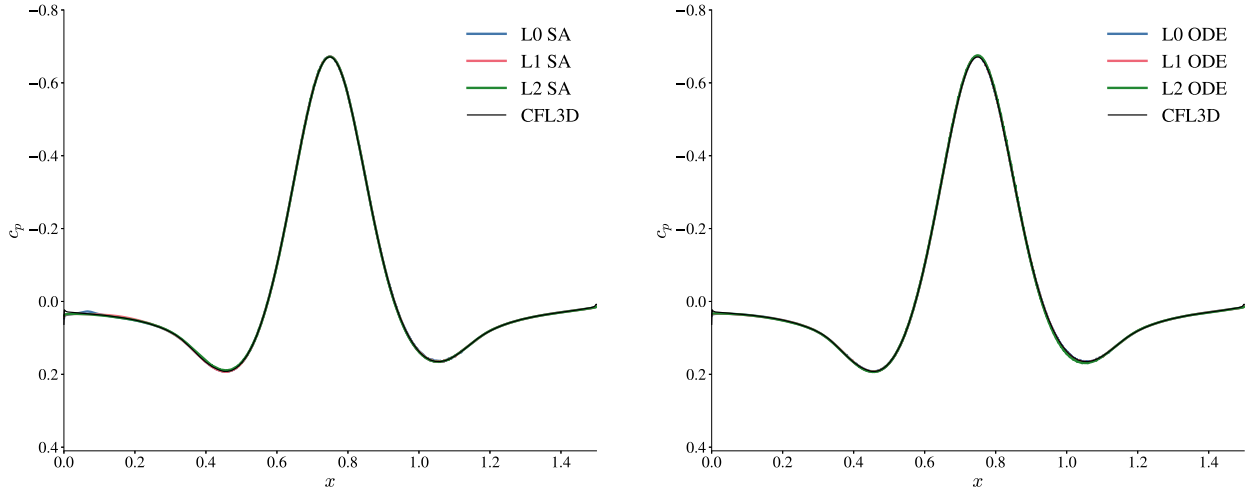
(a) SA wall function at $x = 0.75$

(b) SA wall function at $x = 1.20148$

(c) ODE wall function at $x = 0.75$

(d) ODE wall function at $x = 1.20148$

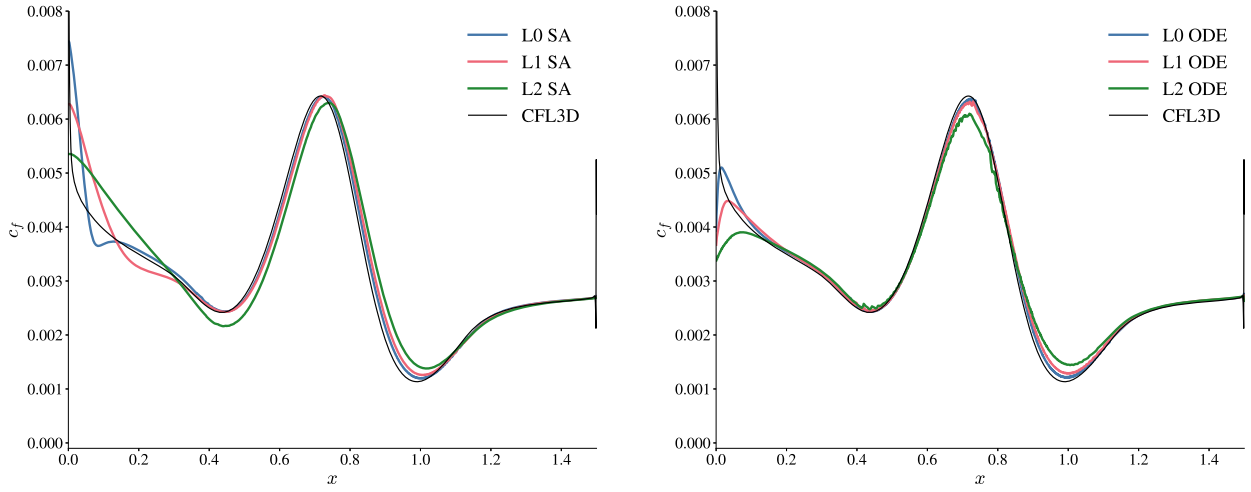
Fig. 16. Velocity profiles for the 2D bump case. Both the SA and the ODE wall functions match the CFL3D solution data.



(a) SA Analytical Wall Function

(b) ODE Wall Function

Fig. 17. The pressure coefficients for the 2D bump case. The pressure matches the CFL3D solution well on all levels of meshes.



(a) SA Analytical Wall Function (b) ODE Wall Function

Fig. 18. The skin frictions for the 2D bump case match the CFL3D solutions. Each wall function exhibits convergence towards the expected solution as the mesh is refined. The effect of the pressure oscillations in the solution is clear in the ODE.

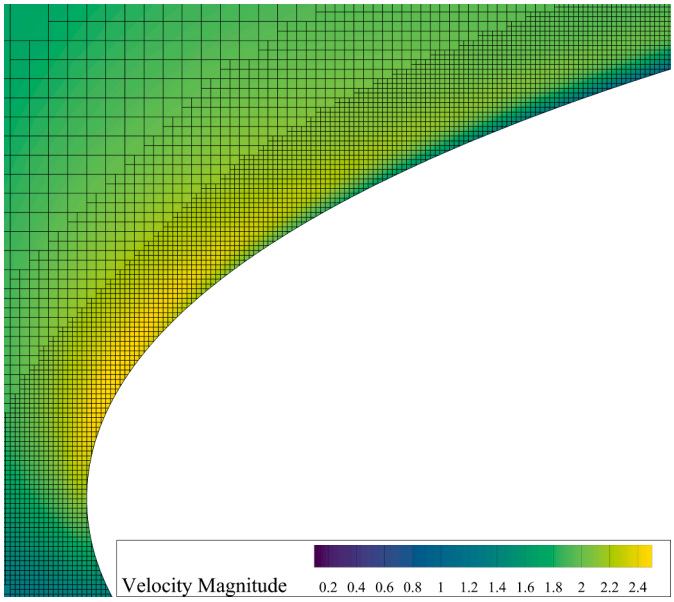
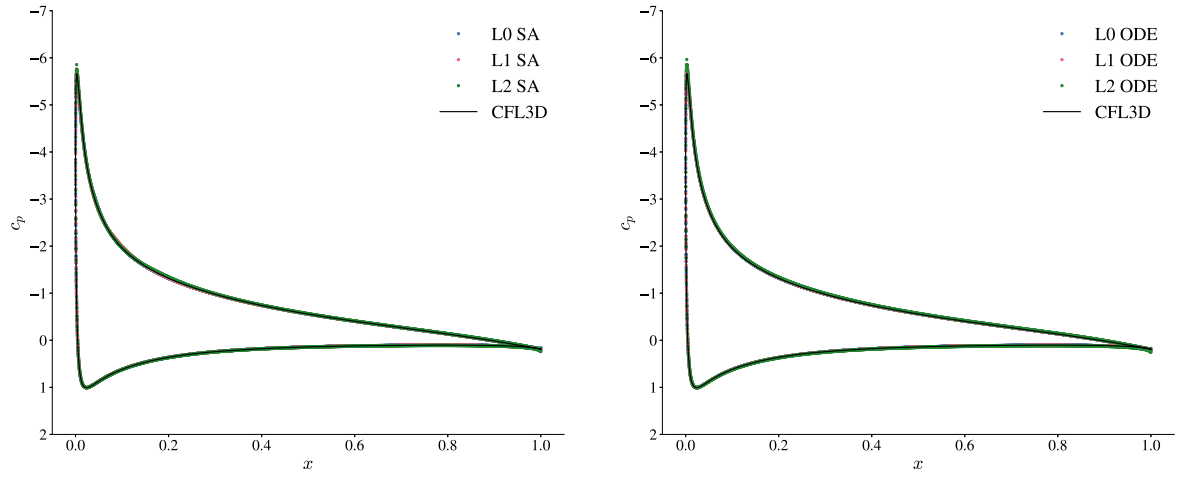
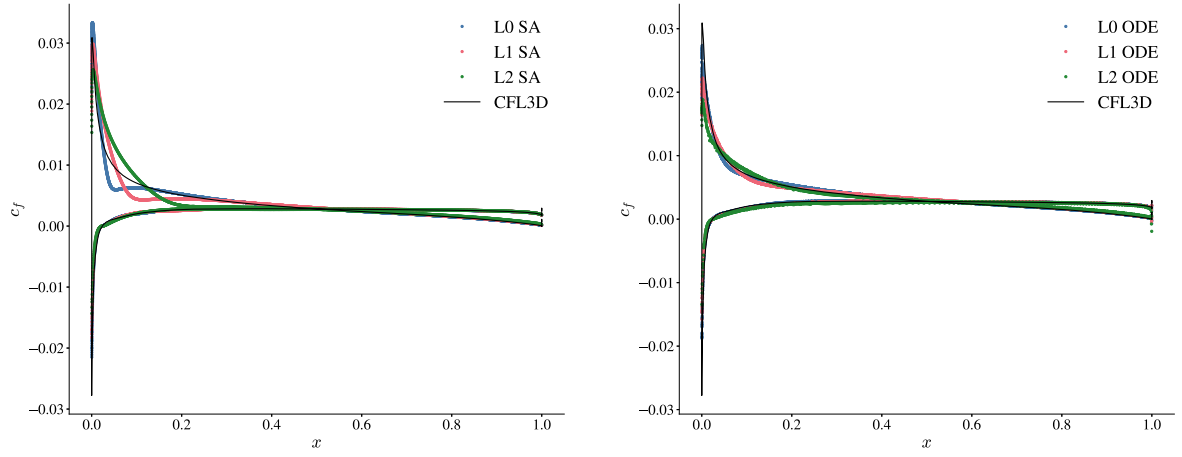


Fig. 19. The L1 mesh for the NACA 0012 case. This image shows the leading edge and suction peak. The suction peak is not resolved enough to capture velocities lower than free stream in the boundary layer.



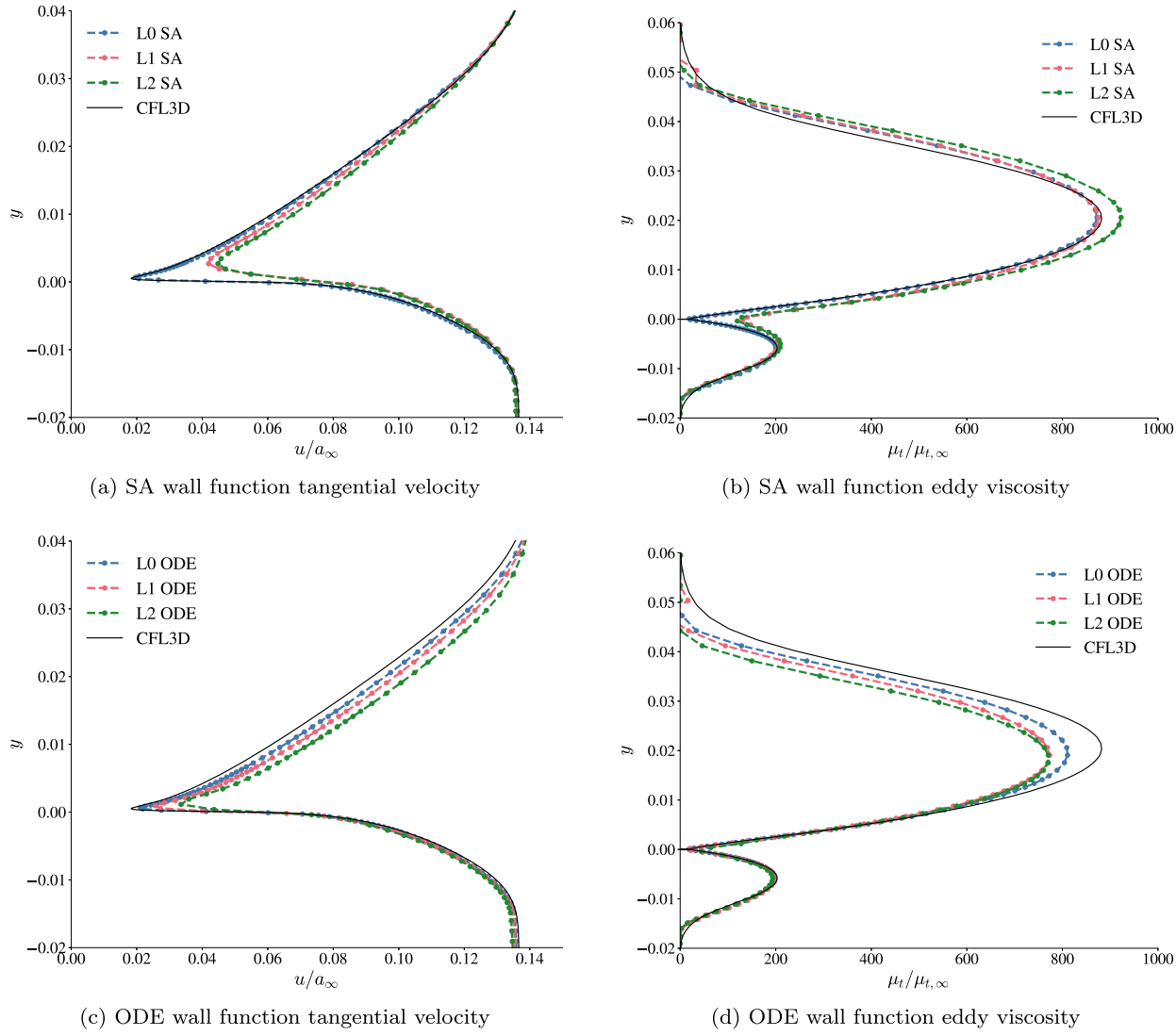
(a) SA wall function (b) ODE wall function

Fig. 20. The pressure coefficients for the NACA 0012 airfoil case. The pressures match the CFL3D solution well on all levels of meshes for both wall functions.



(a) SA wall function (b) ODE wall function

Fig. 21. The skin friction coefficients for the NACA 0012 airfoil case. The SA wall function skin frictions struggle to match the skin frictions in the presence of the strong pressure gradient near the suction peak. The ODE wall function is able to handle the suction peak pressure gradient much better.



(a) SA wall function tangential velocity

(b) SA wall function eddy viscosity

(c) ODE wall function tangential velocity

(d) ODE wall function eddy viscosity

Fig. 22. The tangential velocity and eddy viscosity profiles at the trailing edge of the airfoil. Both methods are able to closely follow the troughs in both quantities near the wall.

5. Conclusion

We implemented an ODE wall function with pressure-momentum balance for high Reynolds number fully turbulent flows on Cartesian cut-cell meshes. We compared the effectiveness of this wall function to an SA analytical wall function and a wall modeled RANS approach. The ODE wall function significantly outperforms traditional approaches by enabling accurate solutions with forcing points well into the wake region ($y^+ > 600$) while maintaining robustness in strong pressure gradients.

Our comparative analysis of three wall-modeling approaches reveals distinct performance characteristics. The analytical SA wall function provides reliable results with forcing points within the log region but struggles with pressure gradients, requiring substantially finer meshes to achieve mesh convergence on cases like the NACA 0012. The WMRANS approach, while implementation friendly, consistently requires higher resolution than wall function methods and exhibits poor performance on irregular cut-cell meshes. In contrast, the ODE wall function demonstrates superior accuracy across all test cases, particularly in strong pressure gradient fields, where it achieves correct skin friction predictions on meshes coarser than those required by the SA wall function. This wall function clearly demonstrates the advantage that modeling the pressure-momentum balance through the boundary layer can have.

The novel ODE wall function implementation incorporates three important pieces: (1) a robust Newton–Krylov solver with pseudo-transient continuation for the 1D ODE system, (2) a MOESS-based adaptive mesh refinement for optimal degree of freedom distribution, and (3) a comprehensive forcing point reconstruction strategy using inverse distance weighting. These components work synergistically to create a computationally efficient method that solves systems with hundreds of degrees of freedom in fewer than ten linear iterations per ODE.

The primary limitation of the approach is that it is sensitive to pressure gradient oscillations in cut-cell regions, manifesting as skin friction noise in the ODE results. The oscillations come from interpolation artifacts near irregular cut-cell boundaries and can be addressed through enhanced reconstruction methods, improved coupling strategies, or alternative stencil selection algorithms. Our current interpolation approach uses two different interpolation schemes for the primal and gradient interpolated values. The SA wall function avoids these oscillations by depending solely on tangential velocity data with no gradients. The WMRANS and the ODE method are sensitive to gradient quantities near cut cells.

This work establishes a clear pathway toward practical high Reynolds number turbulent flows using RANS simulations on automatically generated meshes. Demonstrating this capability enables an ap-

proach for automated aerodynamic design workflows in two dimensions. Given the robust 1D ODE framework developed here, the three-dimensional application extension is straightforward.

The broader applications of the proposed approach extend beyond traditional CFD applications. This capability could enable entirely new fields of study, such as topology optimization for complex aerodynamic flows, real-time design space exploration, and high-throughput aerodynamic database generation by eliminating the mesh generation bottleneck currently dominating aerodynamic analysis workflows. Combining automatic meshing with robust high Reynolds number flow prediction brings computational aerodynamics closer to the "push-button" analysis tools that design engineers require.

This work was supported by the National Science Foundation Graduate Research Fellowship under Grant No. (DGE 2241144).

CRediT authorship contribution statement

Alex Kleb: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Funding acquisition, Conceptualization; **Krzysztof J. Fidkowski:** Writing – review & editing, Supervision, Methodology; **Joaquim R. R. A. Martins:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Alex Kleb reports financial support was provided by National Science Foundation. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Jameson A, Martinelli L, Pierce NA. Optimum aerodynamic design using the Navier–Stokes equations. Theor. Comput. Fluid Dyn. 1998;10(1–4):213–37.
[2] Anderson WK, Venkatakrishnan V. Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation. Comput. Fluids 1999;28(4):443–80.
[3] Nielsen EJ, Anderson WK. Aerodynamic design optimization on unstructured meshes using the Navier–Stokes equations. AIAA J. 1999;37(11):1411–19.
[4] Jameson A, Martinelli L, Alonso JJ, Vassberg JC, Reuther J. Computational Fluid Dynamics for the 21st Century; chap. Perspectives on Simulation Based Aerodynamic Design. Berlin, Heidelberg: Springer. ISBN 978-3-540-44959-1; 2001, p. 135–78.
[5] Martins J RRA. Aerodynamic design optimization: challenges and perspectives. Comput. Fluids 2022;239:105391.
[6] Slotnick J, Khodadoust A, Alonso J, Darmofal D, Gropp W, Lurie E, et al. CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences. Tech. Rep. CR–2014-218178; NASA; 2014.
[7] Aftosmis MJ. Lecture notes in solution adaptive Cartesian grid methods for aerodynamic flows with complex geometries. VKI Lect. Ser. 1997.
[8] Aftosmis MJ, Berger MJ, Melton JE. Robust and efficient Cartesian mesh generation for component-based geometry. AIAA J. 1998;36(6):952–60.
[9] Coirier WJ, Powell KG. Solution-adaptive Cartesian cell approach for viscous and inviscid flows. AIAA J. 1996;34(5):938–45.
[10] Zhao H, Hu P, Kamakoti R, Dittakavi N, Aftosmis M, Marshall D, et al. 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. 2010, p. 1472.
[11] Ye T, Mittal R, Udaykumar HS, Shyy W. An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries. J. Comput. Phys. 1999;156(2):209–40.
[12] Berger M, Aftosmis M. 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. 2012, p. 1301.
[13] Capizzano F. Turbulent wall model for immersed boundary methods. AIAA J. 2011;49(11):2367–81.
[14] Cai S-G, Degrigny J, Boussuge J-F, Sagaut P. Coupling of turbulence wall models and immersed boundaries on Cartesian grids. J. Comput. Phys. 2021;429:109995. .
[15] Kleb A, Fidkowski KJ, Martins J RRA. Development of a Cartesian cut-cell solver for viscous flows. In: AIAA SciTech Forum. 2023,.
[16] Berger MJ, Aftosmis MJ. An ODE-based wall model for turbulent flow simulations. AIAA J. 2018;56(2):700–14.
[17] Ursachi C-I, Allmaras SR, Darmofal DL, Galbraith MC. Stress-equivalent Spalart–Allmaras wall model with local boundary conditions for Reynolds-averaged Navier–Stokes. AIAA J. 2024;62(8):2814–30.
[18] Tamaki Y, Harada M, Imamura T. Near-wall modification of Spalart–Allmaras turbulence model for immersed boundary method. AIAA J. 2017;55(9):3027–39.
[19] Rumsey C. NASA turbulence modeling resource. 2019. Accessed: 2019-03-27.
[20] Spalart P, Allmaras S. A one-equation turbulence model for aerodynamic flows. La Recherche Aerospatiale 1994;1:5–21.
[21] Allmaras SR, Johnson FT, Spalart PR. Modifications and clarifications for the implementation of the Spalart-Allmaras turbulence model. In: Seventh International Conference on Computational Fluid Dynamics. Big Island, Hawaii; 2012.
[22] Kleb A, Fidkowski KJ, Martins J RRA. Extension of a viscous Cartesian cut-cell solver to the compressible RANS equations. In: AIAA Aviation Forum. San Diego, CA; 2023,.
[23] Ceze M, Fidkowski K. 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; 2013.
[24] Pandya MJ, Diskin B, Thomas JL, Frink NT. Improved convergence and robustness of USM3D solutions on mixed-element grids. AIAA J. 2016;54(9):2589–610.
[25] Yildirim A, Kenway G KW, Mader CA, Martins J RRA. A Jacobian-free approximate Newton–Krylov startup strategy for RANS simulations. J. Comput. Phys. 2019;397:108741.
[26] Saad Y. Iterative Methods for Sparse Linear Systems. Society for Industrial and Applied Mathematics; 2003.
[27] Brown PN, Saad Y. Hybrid Krylov methods for nonlinear systems of equations. SIAM J. Sci. Stat. Comput. 1990;11(3):450–81.
[28] Yano M, Darmofal DL. An optimization-based framework for anisotropic simplex mesh adaptation. J. Comput. Phys. 2012;231(22):7626–49.
[29] Ibanez D, Barral N, Krakos J, Loseille A, Michal T, Park M. First benchmark of the unstructured grid adaptation working group. Procedia Eng. 2017;203:154–66.
[30] Ceze MA, Fidkowski KJ. High-order output-based adaptive simulations of turbulent flow in two dimensions. AIAA J. 2016;54(9).