A parallel Newton-Krylov-Schur flow solver for the Navier-Stokes equations discretized using summation-by-parts operators

Michal Osusky[1] and David W. Zingg[2]
University of Toronto, Toronto, Ontario, M3H 5T6, Canada

The objective of the present paper is to demonstrate the effectiveness of a spatial discretization based on summation-by-parts (SBP) operators with simultaneous approximation terms (SATs) in combination with a parallel Newton-Krylov-Schur algorithm for solving the three-dimensional Reynolds-averaged Navier-Stokes equations coupled with the Spalart-Allmaras one-equation turbulence model. The algorithm employs second-order SBP operators on multi-block structured grids with SATs to enforce block interface coupling and boundary conditions. The discrete equations are solved iteratively with an inexact-Newton method, while the linear system at each Newton iteration is solved using a flexible Krylov subspace iterative method with an approximate-Schur parallel preconditioner. The algorithm is verified and validated through the solution of two-dimensional model problems, highlighting the correspondence of the current algorithm with several established flow solvers. A transonic solution over the ONERA M6 wing on a mesh with 15.1 million nodes shows good agreement with experiment. Using 128 processors, the residual is reduced by twelve orders of magnitude in 86 minutes. The solution of transonic flow over the Common Research Model wing-body geometry exhibits the expected grid convergence behavior. The algorithm performs well in solving flows around non-planar geometries and flows with explicitly specified laminar-to-turbulent transition locations. Parallel scaling studies highlight the excellent scaling characteristics of the algorithm on cases with up to 6656 processors and grids with over 150 million nodes.

Nomenclature

A	local (node) flow Jacobian
$\mathcal{A}$	global flow Jacobian
C_D, C_d	coefficient of drag
C_L, C_l	coefficient of lift
C_M	coefficient of moment
C_p, C_f	coefficients of pressure and friction
d	distance to closest surface boundary
D_1, D_2	SBP operators for first and second derivative
e	energy
$\hat{\mathbf{E}}, \hat{\mathbf{F}}, \hat{\mathbf{G}}$	inviscid flux vectors
$\hat{\mathbf{E}}_\mathbf{v}, \hat{\mathbf{F}}_\mathbf{v}, \hat{\mathbf{G}}_\mathbf{v}$	viscous flux vectors
$\mathcal{I}$	identity matrix
J	metric Jacobian
l	reference chord length
M	Mach number
N	number of nodes in computational domain

[1] Ph.D. Candidate, Institute for Aerospace Studies. AIAA Student Member.
[2] Professor and Director, Tier 1 Canada Research Chair in Computational Fluid Dynamics and Environmentally Friendly Aircraft Design, J. Armand Bombardier Foundation Chair in Aerospace Flight, AIAA Associate Fellow.

p	pressure
Pr, Pr_t	laminar and turbulent Prandtl number
$\mathbf{Q}$	conservative flow variables at individual node
$\boldsymbol{\mathcal{Q}}$	conservative flow vector for computational domain
$\boldsymbol{\mathcal{R}}$	flow residual vector for computational domain
Re	Reynolds number
S^*	Sutherland's law constant (198.6°R)
$S_\mathrm{r}, S_\mathrm{c}, S_\mathrm{a}$	row and column scaling matrices
$S, \tilde{S}$	vorticity and vorticity-like term
Δt	time step value
T	temperature
u, v, w	Cartesian velocities
U, V, W	contravariant velocities
x, y, z	Cartesian coordinate axes
γ	specific heat
ϵ	Frechet derivative perturbation parameter
κ_2, κ_4	dissipation coefficients
μ, μ_t	laminar and turbulent viscosity
$\tilde{\nu}$	turbulence variable
ρ	density
σ	dissipation lumping factor
τ_{ij}	viscous stress tensor
ξ, η, ζ	computational coordinate axes

I. Introduction

RECENT advances in computer architecture and numerical methods have paved the way for massively parallel computational infrastructure. Leveraging the ever-increasing access to supercomputer clusters with excess of tens of thousands of processors, large-scale CFD simulations are becoming more suitable for dealing with problems of practical interest. However, accurate simulations of such flows necessitate the solution of very large problems, with the results from the AIAA Drag Prediction Workshop [1] indicating that grids with over O(10^8) grid nodes are required for grid-converged lift and drag values for flows over a wing-body configuration, with the 5th workshop of the series making use of grids with up to 150 million nodes. Following this trend, algorithms that scale well with thousands of processors are required to make efficient use of computational resources.

Established Reynolds-averaged Navier-Stokes (RANS) solvers, such as OVERFLOW [2], FUN3D [3], Flo3xx [4], and NSU3D [5], implement a variety of numerical approaches. These include the use of structured or unstructured grids, finite-volume, finite-element, or finite-difference approximations, explicit or implicit solution strategies, and a wide range of linear solvers and preconditioners. Newton-Krylov methods have been shown to solve turbulent flow problems efficiently in serial implementations [6, 7]. Wong and Zingg [8] applied the Newton-Krylov method to the solution of three-dimensional aerodynamic flows on unstructured grids. This paper presents an efficient parallel three-dimensional multi-block structured solver for turbulent flows over aerodynamic geometries, extending previous work [9, 10] on an efficient parallel Newton-Krylov flow solver for the Euler equations and the Navier-Stokes equations in the laminar flow regime. The combination of techniques employed in the current solver also has the benefit of lending itself to a unified approach for performing both steady and implicit unsteady computations [11]. An efficient and robust Newton-Krylov flow solver can readily serve as the core of an aerodynamic optimization algorithm, as was demonstrated by Nemec and Zingg [12] for two-dimensional turbulent flow and by Hicken and Zingg [13] for three-dimensional inviscid flow. More recent work by Osusky and Zingg [14] has made use of the current flow solution algorithm for three-dimensional aerodynamic shape optimization in the turbulent flow regime.

The choice of spatial discretization method and grid type is heavily influenced by the class of flows and the complexity of geometries the algorithm is to be applied to; the current algorithm is intended for solving compressible turbulent flows around clean geometries, either for analysis or optimization. For smooth geometries, Shu [15] states that finite-difference methods are more efficient in terms of computational cost and programming complexity than finite-volume or discontinuous-Galerkin methods. Furthermore, finite-difference methods extend to higher-order in a straightforward and efficient manner. Structured multi-block grids provide a relatively straightforward means of creating meshes around complex three-dimensional geometries. By breaking the computational domain into several subdomains, or blocks, the approach lends itself readily for use with parallel solution algorithms. The grids can also easily provide higher resolution of boundary layers without adversely affecting the grid spacing elsewhere in the domain. Grids created using this method, however, introduce block interfaces into the computational domain. The treatment of the spatial discretization at interfaces can impact the efficiency of an algorithm, especially in large parallel applications. In the current algorithm, simultaneous approximation terms (SATs) are used to impose boundary conditions, as well as inter-block solution coupling, through a penalty method approach. SATs were originally introduced to treat boundary conditions in an accurate and time-stable manner [16], and later extended to deal with block interfaces [17–19]. Svärd *et al.* [20, 21] and Nordström *et al.* [22] have shown the application of SATs for the Navier-Stokes equations to unsteady problems, as well as some steady model problems. This approach has several advantages over more traditional approaches: it eliminates the need for mesh continuity across block interfaces, further simplifying mesh generation; it reduces the communication for parallel algorithms, especially when extended to higher-order discretizations; and it ensures linear time stability when coupled with summation-by-parts (SBP) operators. SBP operators provide finite-difference approximations to derivatives, while also presenting a systematic means of deriving higher-order operators with a stable and suitably high-order boundary treatment. Consequently, a high-order SBP-SAT finite-difference discretization is a promising alternative to other high-order approaches, such as finite-volume and discontinuous-Galerkin methods. However, the SBP-SAT approach has received limited use in computational aerodynamics applications, in part because SATs present a difficulty in that they can necessitate the use of small time steps with explicit solvers [23]. Hence, the combination of SATs with a parallel Newton-Krylov solver has the potential to be an efficient approach, as demonstrated by Hicken and Zingg [9].

Parallel preconditioning is a critical component of a scalable Newton-Krylov algorithm. Hicken *et al.* [24] have shown that an approximate-Schur preconditioner scales well to at least 1000 processors when inviscid and laminar flows are considered. The objective of the present paper is to demonstrate the applicability of a spatial discretization based on the SBP-SAT approach with a parallel Newton-Krylov-Schur algorithm to the Reynolds-averaged Navier-Stokes equations coupled with the Spalart-Allmaras one-equation turbulence model [25], resulting in a novel and powerful solver for turbulent flows. Although the algorithm presented is second-order in space, with the exception of the convective terms in the turbulence model, it can be efficiently extended to higher-order [26].

The paper is divided into the following sections. Section II presents a brief overview of the governing equations, while Section III presents the spatial discretization used. Section IV provides details of the Newton-Krylov-Schur method and its application to solving the large nonlinear system resulting from the discretization of the governing equations. Section V presents results obtained with the current algorithm for steady flow solutions, including transonic flow solutions around the ONERA M6 and the NASA Common Research Model (CRM) geometry, explicitly tripped flow, solutions on non-planar geometries, and parallel scaling performance characteristics of the current algorithm. Conclusions are given in Section VI.

II. Governing Equations

A. The Navier-Stokes Equations

The three-dimensional Navier-Stokes equations, after the coordinate transformation $(x, y, z) \rightarrow (\xi, \eta, \zeta)$, are given by

$$\partial_t \hat{\mathbf{Q}} + \partial_\xi \hat{\mathbf{E}} + \partial_\eta \hat{\mathbf{F}} + \partial_\zeta \hat{\mathbf{G}} = \frac{1}{Re}\Big(\partial_\xi \hat{\mathbf{E}}_\mathbf{v} + \partial_\eta \hat{\mathbf{F}}_\mathbf{v} + \partial_\zeta \hat{\mathbf{G}}_\mathbf{v}\Big), \tag{1}$$

where

$$\begin{aligned}
&\hat{\mathbf{Q}} = J^{-1}\mathbf{Q}, \\
\hat{\mathbf{E}} &= J^{-1}\Big(\xi_x\mathbf{E} + \xi_y\mathbf{F} + \xi_z\mathbf{G}\Big), \quad \hat{\mathbf{E}}_\mathbf{v} = J^{-1}\Big(\xi_x\mathbf{E}_\mathbf{v} + \xi_y\mathbf{F}_\mathbf{v} + \xi_z\mathbf{G}_\mathbf{v}\Big), \\
\hat{\mathbf{F}} &= J^{-1}\Big(\eta_x\mathbf{E} + \eta_y\mathbf{F} + \eta_z\mathbf{G}\Big), \quad \hat{\mathbf{F}}_\mathbf{v} = J^{-1}\Big(\eta_x\mathbf{E}_\mathbf{v} + \eta_y\mathbf{F}_\mathbf{v} + \eta_z\mathbf{G}_\mathbf{v}\Big), \\
\hat{\mathbf{G}} &= J^{-1}\Big(\zeta_x\mathbf{E} + \zeta_y\mathbf{F} + \zeta_z\mathbf{G}\Big), \quad \hat{\mathbf{G}}_\mathbf{v} = J^{-1}\Big(\zeta_x\mathbf{E}_\mathbf{v} + \zeta_y\mathbf{F}_\mathbf{v} + \zeta_z\mathbf{G}_\mathbf{v}\Big),
\end{aligned}$$

$Re = \frac{\rho_\infty a_\infty l}{\mu_\infty}$, '$a$' is the sound speed, and J is the metric Jacobian that results from the coordinate transformation. The '∞' subscript denotes a free-stream value for the given quantity. The notation ξ_x, for example, is a shorthand form of $\partial_x \xi = \frac{\partial \xi}{\partial x}$. The conservative variables and inviscid and viscous fluxes are given by

$$\mathbf{Q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{bmatrix}, \quad \mathbf{E} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u(e+p) \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ v(e+p) \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} \rho v \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ w(e+p) \end{bmatrix},$$

$$\mathbf{E_v} = \begin{bmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ E_{v,5} \end{bmatrix}, \quad \mathbf{F_v} = \begin{bmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ \tau_{yz} \\ F_{v,5} \end{bmatrix}, \quad \mathbf{G_v} = \begin{bmatrix} 0 \\ \tau_{zx} \\ \tau_{zy} \\ \tau_{zz} \\ G_{v,5} \end{bmatrix}.$$

The preceding variables have been made dimensionless by the use of the free-stream values of density and sound speed, as well as chord length. Laminar viscosity is calculated using a dimensionless form of Sutherland's law [27]:

$$\mu = \frac{a^3(1 + S^*/T_\infty)}{a^2 + S^*/T_\infty}, \tag{2}$$

where T_∞ is typically set to 460°R.

B. Spalart-Allmaras One-Equation Turbulence Model

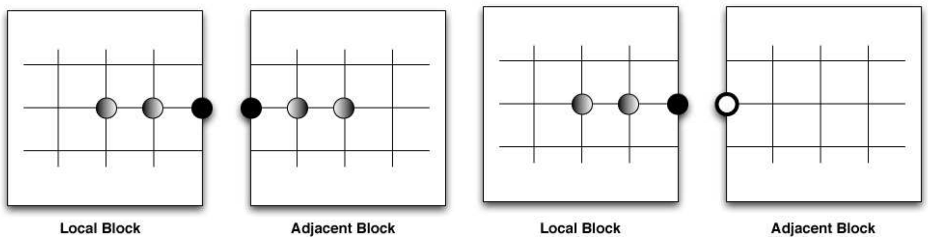
In order to solve turbulent flows, a turbulent, or eddy, viscosity, μ_t, can be added to the viscosity, μ. The Spalart-Allmaras one-equation turbulence model [25] is used to compute the turbulent viscosity. The model solves a transport equation for a turbulence variable, $\tilde{\nu}$, that is related to turbulent viscosity. The model itself is a sixth equation that is solved concurrently with the five Navier-Stokes equations.

The standard version of the model is used in this work, given by

$$\begin{aligned} \frac{\partial \tilde{\nu}}{\partial t} + u_i \frac{\partial \tilde{\nu}}{\partial x_i} = \frac{c_{b1}}{Re}[1 - f_{t2}]\tilde{S}\tilde{\nu} + \frac{1 + c_{b2}}{\sigma_t Re}\nabla \cdot [(\nu + \tilde{\nu})\nabla\tilde{\nu}] - \frac{c_{b2}}{\sigma_t Re}(\nu + \tilde{\nu})\nabla^2\tilde{\nu} \\ - \frac{1}{Re}\left[c_{w1} f_w - \frac{c_{b1}}{\kappa^2} f_{t2}\right]\left(\frac{\tilde{\nu}}{d}\right)^2 + Re f_{t1} \Delta U^2, \end{aligned} \tag{3}$$

where $\nu = \frac{\mu}{\rho}$. The spatial derivatives on the left side of the equation represent advection. The first term on the right side represents production, while the second and third terms account for diffusion. The fourth term represents destruction. The final term allows for the specification of an explicit laminar-turbulent transition location, but it should be stressed that the model cannot predict this location; it has to be specified either by the user or some other means, such as the e^N method [28]. For fully turbulent flows, the value f_{t1} is set to zero, omitting the explicit trip terms. Refer to [25] for the values of all the terms and constants that appear in Eq. (3). Importantly, precautions are taken to ensure that neither the vorticity, S, nor the vorticity-like term, $\tilde{S}$, approaches zero or becomes negative, which could lead to numerical problems.

As an alternative to trimming negative turbulence values, the presence of which would result in numerical problems for the above model, Allmaras *et al.* present a "negative" variant of the model in [29]. The model variant accepts negative values of $\tilde{\nu}$ and modifies the diffusive, production, and destruction terms for such instances. However, in the present work, it was not found to provide an advantage in terms of robustness. One of the reasons for this is that the current discretization does not result in negative turbulence quantities in the converged solution and trimming is adequate in dealing with any negative values that occur during convergence to steady state.



a) Typical discretization **b) SBP-SAT discretization**

Fig. 1 Node information required for spatial discretization at block interface

III. Spatial Discretization

The spatial discretization of the Navier-Stokes equations and the turbulence model is obtained by the use of Summation-By-Parts (SBP) operators, while inter-block coupling and boundary conditions are enforced weakly by the use of Simultaneous Approximation Terms (SATs). We decompose our domains into multiple blocks, resulting in multi-block structured grids. This blocking is conducive to localization of high grid density in certain areas without affecting the grid resolution elsewhere, which can be especially effective when resolving turbulent boundary layers or shocks; individual blocks with high node density can be placed where required.

One of the important advantages of SATs lies in the manner in which they allow the algorithm to deal with both domain boundaries and block interfaces. This is illustrated in Fig. 1, which compares how the governing equations would be handled at an interface with a typical discretization and the current SBP-SAT discretization. The typical spatial discretization in Fig. 1a) requires spatial derivatives to be formed across the interface. This is achieved by the use of halo, or ghost, nodes, effectively completing a full internal discretization stencil with information from the adjoining block. The SBP-SAT approach, illustrated in Fig. 1b), instead forms a local one-sided approximation to the required derivatives. Additionally, the SAT itself is added, comprised of a difference between the flow variables on the local node and the corresponding node on the adjoining block. When dealing with viscous terms for the Navier-Stokes equations and the diffusive terms for the Spalart-Allmaras turbulence model, differences in fluxes also have to be considered. Although this approach does not result in a transparent interface treatment, the error introduced at the interfaces is small and consistent with the order of accuracy of the method [22, 30].

The SBP-SAT approach requires less information from adjoining blocks in order to obtain a discretization of the governing equations at block interfaces. This results in a reduced requirement for information sharing between blocks, which is especially advantageous for parallel algorithms, reducing the time spent in communication. Additionally, the fact that this discretization does not need to form any derivatives across interfaces reduces the continuity requirements for mesh generation at interfaces. In fact, only C^0 continuity is necessary, allowing the algorithm to provide accurate solutions even on grids with relatively high incidence angles for grid lines at interfaces, an example of which can be seen in Fig. 2. Using a typical discretization approach on such a grid would result in errors in any spatial derivatives taken across the interfaces. This feature substantially reduces the burden placed on the grid generation process, especially for complex three-dimensional geometries, such as the NASA Common Research Model (CRM) [31].

This section presents the SBP operators for first and second derivatives and their application to the governing equations, as well as the various SATs required. The focus of this paper is to present the implementation of the discretization for the viscous terms and turbulence model, with the details of the Euler equation implementation presented previously by Hicken and Zingg [9]. Additional details, including the theory behind the development of the SBP-SAT approach, can be found in references [9, 10, 20–22, 32], and [33]. It is important to note that while the algorithm presented in this paper is second-order accurate, the SBP-SAT discretization provides a relatively straightforward approach to extending to higher orders of accuracy [11, 26, 34].

Numerical dissipation is added using either the scalar dissipation model developed by Jameson *et al.* [35] and later refined by Pulliam [36], or the matrix dissipation model of Swanson and Turkel [37]. With the current second-order spatial discretization (with the exception of the convective

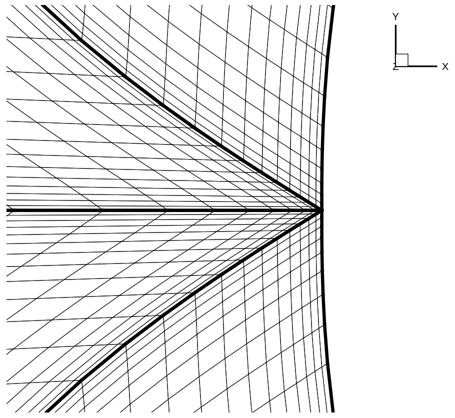


Fig. 2 Blocks and grid-lines at leading edge of wing in an H-H topology grid with C^0 continuity

terms in the turbulence model), the numerical dissipation consists of second- and fourth-difference dissipation operators, whose magnitudes are controlled by the κ_2 and κ_4 coefficients, respectively, typically set to 2.0 and 0.04 for transonic flows. For subsonic flows, κ_2 is set to 0.

Grid metrics, which result from the coordinate transformation, are computed using the second-order SBP operator for a first derivative,

A. Summation-by-parts operators

SBP operators allow for the construction of finite-difference approximations to derivatives. When dealing with the governing equations considered in this work, approximations to both the first and second derivatives are required.

The SBP operators for the first derivative were originally derived by Kreiss and Scherer [32], subsequently extended by Strand [33], and applied by various authors (see [9, 22, 26, 38, 39]). SBP operators are centered difference schemes that do not include boundary conditions; in our case these are enforced using SATs. They are constructed so that the discrete energy-method can be used to make time stability statements about a discretization and have been shown to be time-stable for the linearized Navier-Stokes equations [38]. For curvilinear coordinates, however, time-stability can only be guaranteed for SBP operators constructed with a diagonal norm. Hence, this type of operator is considered for this work.

1. SBP operator for first derivative

In this section we briefly present the second-order SBP operator for the first derivative. A globally second-order accurate operator for a first derivative is given by

$$D_1 = H^{-1}\Theta, \tag{4}$$

where

$$H = h\begin{bmatrix} \frac{1}{2} & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & \frac{1}{2} \end{bmatrix}, \quad \Theta = \frac{1}{2}\begin{bmatrix} -1 & 1 & & & \\ -1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 0 & 1 \\ & & & -1 & 1 \end{bmatrix},$$

and h takes on the value of the spacing in the pertinent coordinate direction, either $\Delta\xi$, $\Delta\eta$, or $\Delta\zeta$. In the context of the uniform computational grid, h has a value of 1 for all three coordinate directions. The second-order operator takes on the form of a centered difference approximation on the interior of a block, with one-sided first-order treatment at block boundaries.

Application to Navier-Stokes equations

The D_1 operator is used to obtain a finite difference approximation of the inviscid fluxes for the entire computational domain. For example, the inviscid flux in the ξ-direction can be taken as

$$\partial_\xi \hat{\mathbf{E}} \approx D_{1\xi} \hat{\mathbf{E}}. \tag{5}$$

With the viscous component of the Navier-Stokes equations, the D_1 operator is used in the discretization of the cross-derivative terms, which have the form

$$\partial_\xi (\beta \partial_\eta \alpha), \tag{6}$$

where β is a spatially varying coefficient, and α is a flow quantity such as the x-component of velocity, u. Using Eq. (4), the cross-derivative can be approximated as

$$D_{1\xi} \boldsymbol{\beta} D_{1\eta} \boldsymbol{\alpha}, \tag{7}$$

resulting in the following interior discretization (at node (j,k,m)):

$$\frac{1}{2}\beta_{j+1,k,m}\left(\frac{\alpha_{j+1,k+1,m}-\alpha_{j+1,k-1,m}}{2}\right)-\frac{1}{2}\beta_{j-1,k,m}\left(\frac{\alpha_{j-1,k+1,m}-\alpha_{j-1,k-1,m}}{2}\right). \tag{8}$$

An analogous approach is used for terms where cross-derivatives in any two directions are required.

Application to Spalart-Allmaras turbulence model

The advective terms that appear in the turbulence model consist of first derivatives of the turbulence variable, $\tilde{\nu}$, multiplied by velocities. An example of this is the term associated with the spatial derivative in the ξ-direction, given by

$$U \partial_\xi \tilde{\nu}, \tag{9}$$

where U is defined as $\xi_x u + \xi_y v + \xi_z w$.

The authors of the model suggest the use of an upwinding strategy when discretizing this term, which is the approach taken here. However, in the context of SBP operators, we have made use of the connection between upwinding and artificial dissipation, namely that an upwinded operator can be expressed as a centered difference operator added to a dissipative operator. The derivative can be taken as

$$U \partial_\xi \tilde{\nu} \approx \mathbf{U} D_1 \tilde{\boldsymbol{\nu}} + \frac{1}{2} |\mathbf{U}| H^{-1} D_d^T D_d \tilde{\boldsymbol{\nu}} \tag{10}$$

where $\tilde{\boldsymbol{\nu}}$ represents a vector containing the turbulence quantity in the domain, and

$$\mathbf{U} = \mathrm{diag}\left(U_1, U_2, ..., U_N\right), \quad |\mathbf{U}| = \mathrm{diag}\left(|U_1|, |U_2|, ..., |U_N|\right), \quad D_d = \begin{bmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \\ & & & & 0 \end{bmatrix},$$

where N is the number of nodes in the pertinent coordinate direction.

The above SBP discretization provides a clear approach to dealing with block boundaries. For completeness, the following shows the resulting discretization in different parts of the domain:

low side:	$\left(U_1 - \lvert U_1\rvert\right)\left(\tilde{\nu}_2 - \tilde{\nu}_1\right),$
interior:	$\frac{1}{2}U_j\left(\tilde{\nu}_{j+1} - \tilde{\nu}_{j-1}\right) - \frac{1}{2}\lvert U_j\rvert\left(\tilde{\nu}_{j+1} - 2\tilde{\nu}_j + \tilde{\nu}_{j-1}\right),$
high side:	$\left(U_N + \lvert U_N\rvert\right)\left(\tilde{\nu}_N - \tilde{\nu}_{N-1}\right).$

The first derivative operator of Eq. (4) is also used in the discretization of the vorticity term, S.

2. SBP operator for second derivative with variable coefficients

The compressible Navier-Stokes equations require a discrete approximation to derivatives of the form $\partial_\xi(\beta\partial_\xi\alpha)$. The simplest means of discretizing these terms is to apply the first derivative twice. Alternatively, one can construct a discrete approximation that has the same stencil width as the first derivative, called a compact-stencil operator. The application of the first derivative twice has several disadvantages compared to compact-stencil operators: larger bandwidth, loss of one order of accuracy, higher global error, and less dissipation of high wavenumber modes [40]. Given these shortcomings, our approach is to use compact SBP operators for the second derivative with variable coefficients.

The second-order operator, originally developed by Mattsson *et al.* [38], can be expressed as

$$D_2(\beta) = H^{-1}\left\{-\left(D_1\right)^T HBD_1 - \frac{1}{4h}\left(\tilde{D}_2\right)^T C_2 B\tilde{D}_2 + EBD_1^{(2)}\right\}, \tag{11}$$

where

$$\tilde{D}_2 = \begin{bmatrix} 1 & -2 & 1 & & & & \\ 1 & -2 & 1 & & & & \\ & 1 & -2 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 1 & -2 & 1 & \\ & & & & 1 & -2 & 1 \\ & & & & 1 & -2 & 1 \end{bmatrix}, \quad C_2 = \begin{bmatrix} 0 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & 0 \end{bmatrix},$$

$$EBD_1^{(2)} = \frac{1}{h}\begin{bmatrix} \frac{3\beta_1}{2} & -2\beta_1 & \frac{\beta_1}{2} & & \\ 0 & 0 & 0 & & \\ & \ddots & \ddots & \ddots & \\ & & 0 & 0 & 0 \\ & & \frac{\beta_N}{2} & -2\beta_N & \frac{3\beta_N}{2} \end{bmatrix},$$

B is a diagonal matrix containing the spatially varying coefficients, and D_1 is defined in Eq. (4). Where necessary, the notation $D^{(b)}$ provides the order of the operator at block boundaries (b). The tilde symbol signifies an undivided difference operator.

The double-derivative in the viscous terms of the Navier-Stokes equations and the diffusive terms of the turbulence model can be approximated as

$$\partial_\xi\left(\beta\partial_\xi\alpha\right) \approx D_2(\beta)\boldsymbol{\alpha}. \tag{12}$$

For an internal node, this will result in the narrow stencil used by Pulliam [36] (with k and m subscripts suppressed):

$$\frac{1}{2}(\beta_{j+1}+\beta_j)(\alpha_{j+1}-\alpha_j) - \frac{1}{2}(\beta_j+\beta_{j-1})(\alpha_j-\alpha_{j-1}). \tag{13}$$

At block boundaries, where one-sided differences are employed, the discretization takes on the form

$$\begin{array}{ll} \text{low side:} & -\beta_1\left[2(\alpha_2-\alpha_1)-(\alpha_3-\alpha_2)\right]+\beta_2(\alpha_2-\alpha_1), \\ \text{high side:} & \beta_N\left[2(\alpha_N-\alpha_{N-1})-(\alpha_{N-1}-\alpha_{N-2})\right]-\beta_{N-1}(\alpha_N-\alpha_{N-1}). \end{array} \tag{14}$$

B. Simultaneous Approximation Terms

The use of SBP operators ties in closely to the application of SAT penalties at block boundaries, either interfaces or domain boundaries. SATs are used to preserve inter-block continuity, or enforce specific boundary conditions. The purpose of this section is not to derive the forms of the various SATs used, but rather to present the implementation used in the present algorithm. See references [9, 20–22] for an analysis and derivation of the SAT terms applied to the Navier-Stokes equations. All SAT terms that follow are shown in the form in which they would be added to the right-hand-side of the governing equations.

1. SATs for Navier-Stokes equations

The form of the inviscid, or Euler, portion of the SATs on the low side of a block is

$$\mathrm{SAT}_{\mathrm{inv}} = -H_b^{-1} J^{-1} A_\xi^+ \left(\mathbf{Q} - \mathbf{Q}_{\mathbf{target}}\right), \tag{15}$$

where H_b is the boundary node element of the diagonal norm matrix H,

$$A_\xi^+ = \tfrac{A_\xi + |A_\xi|}{2}, \quad A_\xi = \tfrac{\partial \hat{\mathbf{E}}}{\partial \hat{\mathbf{Q}}},$$

and $\mathbf{Q}$ are the flow variables on the boundary node in the current block. $|A_\xi|$ denotes $X^{-1}|\Lambda|X$, where X is the right eigenmatrix of A_ξ, and Λ contains the eigenvalues along its diagonal. At a high-side boundary A_ξ^- is used to capture the incoming characteristics, and the sign of the penalty is reversed. When dealing with boundaries normal to the other two coordinate directions, A_ξ is replaced by either A_η or A_ζ. The variable $\mathbf{Q}_{\mathbf{target}}$ takes on the target values to which the local values of $\mathbf{Q}$ are being forced. When dealing with a block interface, these are the flow variable values on a coincident node in a neighbouring block, or, when dealing with a far-field boundary, they can be the free-stream flow variable values. A number of different boundary conditions, such as a slip-wall or symmetry plane, can be enforced using this approach for the Euler equations. In each case, the SAT works on a principle very similar to characteristic boundary conditions.

The basis of the viscous SATs is presented by Nordström *et al.* [22] and is summarized below, with special attention being paid to each type of block boundary.

The first type of viscous SAT deals with differences in viscous fluxes. In the ξ-direction, this term has the form

$$\mathrm{SAT}_{\mathrm{visc\_flux}} = \frac{H_b^{-1}\sigma^V}{Re}\left(\hat{\mathbf{E}}_{\mathbf{v}} - \hat{\mathbf{E}}_{\mathbf{v},\mathbf{target}}\right), \tag{16}$$

where $\hat{\mathbf{E}}_{\mathbf{v}}$ is the local viscous flux, and $\hat{\mathbf{E}}_{\mathbf{v},\mathbf{target}}$ is the target value of the viscous flux. Additionally, $\sigma^V = 1$ at a low-side boundary, and -1 at a high-side boundary. At a far-field boundary, which is supposed to force the solution towards free-stream conditions, $\hat{\mathbf{E}}_{\mathbf{v},\mathbf{target}} = 0$. Interface SATs also make use of (16), where $\hat{\mathbf{E}}_{\mathbf{v},\mathbf{target}}$ is equal to $\hat{\mathbf{E}}_{\mathbf{v2}}$, the viscous flux on the coincident node in the adjoining block.

A no-slip adiabatic wall boundary condition is enforced with the use of a different type of term, which is again added on top of the Euler SAT. The form of the viscous portion of the no-slip wall SAT for a boundary at the low or high side of a block in the ξ-direction, is

$$\mathrm{SAT}_{\mathrm{visc\_wall,1}} = \frac{H_b^{-1}\sigma^W}{Re} I \left(\mathbf{Q} - \mathbf{Q}_{\mathbf{target}}\right), \tag{17}$$

where I is the identity matrix,

$$\sigma^W \leq -\frac{\xi_x^2 + \xi_y^2 + \xi_z^2}{J}\frac{\mu}{2\rho}\max\left(\frac{\gamma}{Pr}, \frac{5}{3}\right), \text{ and } \mathbf{Q}_{\mathbf{target}} = \left[\rho_1, 0, 0, 0, \frac{\rho_1 T_2}{\gamma(\gamma-1)}\right]^T,$$

where Pr is set to 0.72 and $\gamma = 1.4$ for air. σ^W is calculated based on local values, while $\mathbf{Q}_{\mathbf{target}}$ is constructed in order to enforce an adiabatic no-slip wall boundary condition. The three momentum components are forced toward zero, thus satisfying the no-slip condition, while no condition is enforced on density, since the local value of density, ρ_1, will cancel out in the penalty term. The energy equation has a penalty term applied to it based on the value of the temperature of one node above the boundary, T_2. This approach will result in a zero temperature gradient at the solid boundary, along with a no-slip velocity condition. The use of T_2 to enforce the adiabatic condition relies on the assumption that the grid is perpendicular to the surface of the wing, which may not always be true. The form of the SAT presented in (17) can also be readily used to enforce an isothermal boundary condition. This can be achieved by replacing the T_2 term in $\mathbf{Q}_{\mathbf{target}}$ with the desired wall temperature, T_w, as described in [21]. Unlike a more traditional method of applying the adiabatic no-slip surface condition, the penalty approach presented here does not apply a boundary condition on the equation for the conservation of mass. This is due to the fact that the Navier-Stokes equations are solved on all nodes, including the boundaries, so we do not need to provide an explicit value for all variables at the surface (as required for some traditional methods).

An alternate approach to dealing with the adiabatic condition involves a combination of previously discussed penalty terms. The surface penalty in (17) can be modified to only enforce the no-slip condition, while the viscous flux penalty in (16) can be modified to enforce the zero-temperature gradient necessary for the adiabatic condition. The overall form of this SAT is

$$\mathrm{SAT}_{\mathrm{visc\_wall},2} = \frac{H_b^{-1}}{Re}\left[\sigma^W I\left(\mathbf{Q}-\mathbf{Q}_{\mathbf{target}}\right)+\sigma^V\left(\hat{\mathbf{E}}_{\mathbf{v}}-\hat{\mathbf{E}}_{\mathbf{v},\mathbf{target}}\right)\right], \tag{18}$$

where

$$\mathbf{Q}_{\mathbf{target}} = \left[\rho_1, 0, 0, 0, e_1\right]^T,$$

and $\hat{\mathbf{E}}_{\mathbf{v},\mathbf{target}}$ is identical to $\hat{\mathbf{E}}_{\mathbf{v}}$, except that the temperature derivative terms normal to the wall are set to zero. The local values of density and energy are given by ρ_1 and e_1, respectively, and the coefficients σ^W and σ^V retain their previously defined values. In this way, the first part of the SAT enforces only the no-slip condition, while the second part enforces the adiabatic condition. Since this approach uses the gradients as they appear in the viscous stresses, it makes no assumptions about the grid (whether or not it is perpendicular to the surface), and enforces a more general condition of $\frac{\partial T}{\partial n} = 0$.

Block interfaces are treated in a similar way, but with additional viscous SATs for penalizing differences in conservative variable values. The form used is:

$$\mathrm{SAT}_{\mathrm{visc\_vars}} = -\frac{H_b^{-1}\sigma^{V_2}}{JRe} B_{\mathrm{int},\xi}\left(\mathbf{Q}-\mathbf{Q}_{\mathbf{target}}\right), \tag{19}$$

where

$$\sigma^{V_2} \leq 0.5$$

for stability, and $\mathbf{Q}_{\mathbf{target}}$ is the vector of conservative flow variables on the coincident node in the adjoining block. The B_{int} matrix is related to the viscous Jacobian, and is derived based on Nordström *et al.* [22]. Refer to the Appendix for the complete form.

In order to reduce the size of the computational domain, symmetry boundaries can be imposed. SATs are again used to impose this boundary condition by using (15) to impose a purely tangential flow ($\mathbf{Q}_{\mathbf{target}}$ is constructed in such a way as to force the normal velocity component to zero). In addition, (17) is used to enforce a zero normal gradient in all conservative variables.

The following is used to enforce the inviscid SAT on an outflow boundary in a viscous flow:

$$\mathrm{SAT}_{\mathrm{inv\_outflow}} = \frac{H_b^{-1}\sigma^I}{J} A_\xi^-\left(\mathbf{Q}_{j_{\max}}-\mathbf{Q}_{j_{\max}-1}\right), \tag{20}$$

in which the boundary is assumed to be on the high side of a block in the ξ-direction. The modification is appropriate in dealing with the viscous wake region. The advantage of this approach is that it requires minimal modification to the existing Euler SAT term, which typically uses the free-stream flow conditions instead of $\mathbf{Q}_{j_{\max}-1}$. An alternate approach to dealing with the outflow condition is presented by Svärd *et al.* [20].

2. SATs for turbulence model

The SAT for the advection portion of the turbulence model needs to account for the flow direction in much the same way as the Euler equation SATs. This can be achieved using the following form of the SAT:

$$\mathrm{SAT}_{\mathrm{adv}} = H_b^{-1}\sigma_{\mathrm{a}}\left(\tilde{\nu}-\tilde{\nu}_{\mathrm{target}}\right), \tag{21}$$

where $\tilde{\nu}$ is the local value of the turbulence variable and $\tilde{\nu}_{\mathrm{target}}$ is the target value of the turbulence variable, which can either be specified by a boundary condition or, in the case of a block interface, the corresponding value on an adjoining block. The SAT parameter σ_{a} is constructed so that it accounts for the direction of information propagation in the flow:

$$\sigma_{\text{a}} = -\frac{1}{2}\left[\max\left(|U|, \phi\right) + \delta_{\text{a}} U\right], \tag{22}$$

where δ_{a} is $+1$ on the low side of a block, and -1 on the high side of a block. On an interface all flow related information in σ_{a}, such as the contravariant velocity U, is based on an average velocity between the coincident interface nodes, while at a domain boundary, it is constructed based on local information only. Finally, ϕ is a limiting factor introduced to prevent the SAT from completely disappearing in regions where the value of U goes to zero, such as near a solid surface. Following the work done on the Euler equation SATs, the value of ϕ was chosen to be

$$\phi = V_l\left(|U| + a\sqrt{\xi_x^2 + \xi_y^2 + \xi_z^2}\right), \tag{23}$$

where $V_l = 0.025$, and a is the speed of sound. The quantity appearing in the brackets above is the spectral radius of the inviscid flux Jacobian.

As with the SATs used for the viscous portion of the Navier-Stokes equations, the SATs for the diffusive portion of the turbulence model consist of two parts, one dealing with the difference in the turbulent quantity, the other dealing with the difference in the turbulent quantity gradient.

The diffusive SAT dealing with the difference in gradients of the turbulence variable has the general form

$$\text{SAT}_{\text{diff\_flux}} = H_b^{-1}\sigma_{\text{df}}\left(g - g_{\text{target}}\right), \tag{24}$$

where σ_{df} is $+1$ on the low side of a block and -1 on the high side of a block. Additionally, the turbulence quantity gradients, denoted by g, have the form

$$g = \frac{1}{\sigma_{\text{t}} Re}\left(\nu + \tilde{\nu}\right)\left(\xi_x^2 + \xi_y^2 + \xi_z^2\right)\delta_{\xi}\tilde{\nu}, \tag{25}$$

where $\delta_{\xi}\tilde{\nu}$ is a one-sided first derivative consistent with the definition of the second derivative SBP operator at block boundaries, specified in the $D_1^{(2)}$ matrix of Eq. (11). The parameter σ_{t} is defined as part of the turbulence model with a value of $2/3$. This SAT is applied at the farfield boundary, with the target gradient set to 0, or at block interfaces, with the target gradient calculated based on values at the interface of the adjoining block.

The diffusive SAT that deals with the difference in flow variables has a form analogous to the viscous SAT presented by Nordström *et al.* [22] for the Navier-Stokes equations,

$$\text{SAT}_{\text{diff\_vars}} = -H_b^{-1}\frac{1}{4\sigma_{\text{t}} Re}\sigma_{\text{dv}}\left(\tilde{\nu} - \tilde{\nu}_{\text{target}}\right), \tag{26}$$

where

$$\sigma_{\text{dv}} = \left(\nu + \tilde{\nu}\right)\left(\xi_x^2 + \xi_y^2 + \xi_z^2\right). \tag{27}$$

As with the advective SAT, the value of σ_{dv} is based on a state average when dealing with an interface, or simply the local state when at a domain boundary. Grid metrics are always taken from the local block information. This SAT is applied at block interfaces, wall boundaries (where the target value is 0), and symmetry planes (where the target value is taken from one node inside the boundary).

While the production and destruction terms act as source terms, therefore not necessitating the application of the SBP-SAT approach due to the absence of spatial derivatives, we have found it necessary to add a source term for nodes located directly on the surface of the aerodynamic body. Even though the distance value is zero at a solid boundary, the original reference for the turbulence model [25] presents limiting values for both the production and destruction terms. The values are based on a local surface shear stress, τ_w. The initialization of the flow conditions to a uniform state results in the shear stress being negligible in the early iterations of the solution process, resulting in insignificant limiting values for the production and destruction terms. The lack of source terms for the surface nodes leads to a significant difference in the residual value between the surface nodes and the nodes directly above the surface. This difference can result in large, destabilizing updates to the turbulence variable, often causing the flow solution to diverge.

To mitigate this, a destruction source term is added to all nodes with a zero off-wall distance in order to stabilize the solution in the early stages of convergence. It is calculated using a value of $d = d_{\min}/2$, where $d_{\min}$ is the smallest non-zero off-wall distance in the entire computational domain. The use of this extra source penalty for the surface nodes does not have a significant impact on the converged solution, as it forces the surface $\tilde{\nu}$ towards 0.

The farfield condition used with the turbulence model for fully turbulent flows sets the target farfield value of $\tilde{\nu}$ to 3.0, as suggested by Spalart and Rumsey [41], while flows with an explicit trip location specified use a target farfield value of 0.1. It should be noted that the value for tripped flows is at the high end of the recommended range, but has resulted in more robust algorithm performance. The target surface value of $\tilde{\nu}$ is set to 0.0.

IV. Solution Methodology

Applying the SBP-SAT discretization described in the previous section to the steady Navier-Stokes equations and the Spalart-Allmaras one-equation turbulence model results in a large system of nonlinear equations, defined as

$$\boldsymbol{\mathcal{R}}(\boldsymbol{\mathcal{Q}}) = 0. \tag{28}$$

When time-marched with the implicit Euler time-marching method and a local time linearization, this results in a large system of linear equations of the form [42]

$$\left(\frac{\mathcal{I}}{\Delta t} + \mathcal{A}^{(n)}\right) \Delta\boldsymbol{\mathcal{Q}}^{(n)} = -\boldsymbol{\mathcal{R}}^{(n)}, \tag{29}$$

where n is the outer (nonlinear) iteration index, $\boldsymbol{\mathcal{R}}^{(n)} = \boldsymbol{\mathcal{R}}(\boldsymbol{\mathcal{Q}}^{(n)})$, $\Delta\boldsymbol{\mathcal{Q}}^{(n)} = \boldsymbol{\mathcal{Q}}^{(n+1)} - \boldsymbol{\mathcal{Q}}^{(n)}$, and

$$\mathcal{A}^{(n)} = \frac{\partial \boldsymbol{\mathcal{R}}^{(n)}}{\partial \boldsymbol{\mathcal{Q}}^{(n)}}.$$

In the infinite time step limit, the above describes Newton's method and will converge quadratically if a suitable initial iterate, $\boldsymbol{\mathcal{Q}}^{(0)}$, is known. This initial iterate must be sufficiently close to the solution of (28). Since it is unlikely that any initial guess made for a steady-state solution will satisfy this requirement, the present algorithm makes use of a start-up phase whose purpose it is to find a suitable initial iterate. The following sections describe each of the phases as they apply to the solution of the Navier-Stokes equations. Both phases result in a large set of linear equations at each outer iteration, which are solved to a specified tolerance using the flexible variant of the preconditioned Krylov iterative solver GMRES. To avoid the modification of the residual vector, right preconditioning is used.

A. Approximate-Newton phase

The approximate-Newton method makes use of implicit Euler time-stepping to find a suitable initial iterate for Newton's method. Since we are not interested in a time-accurate solution, some useful modifications can be made. These include a first-order Jacobian matrix and a spatially varying time step.

A first-order approximation to the Jacobian matrix, $\mathcal{A}_1$, has been shown to be an effective replacement for the true Jacobian, $\mathcal{A}$, during the start-up phase [6, 43, 44]. A number of approximations are made when creating the first-order approximation to the Jacobian. When dealing with the inviscid terms, the fourth-difference dissipation coefficient, κ_4, is combined with the second-difference dissipation coefficient, κ_2, to form a modified second-difference dissipation coefficient, $\tilde{\kappa}_2$, such that

$$\tilde{\kappa}_2 = \kappa_2 + \sigma\kappa_4,$$

A value of $\sigma = 8$ is used with scalar dissipation, while solutions with matrix dissipation use $\sigma = 12$. The modified fourth-difference dissipation coefficient, $\tilde{\kappa}_4$, is set to zero. Applying this lumping approach reduces the number of matrix entries for the inviscid terms, reducing the memory requirements for the code.

The viscous terms, however, still possess a relatively large stencil. To mitigate this, the cross-derivative terms that appear in the viscous stresses are dropped when constructing the first-order Jacobian. This approach reduces the stencil of all interior nodes to nearest neighbors only, matching the stencil size of the inviscid terms, which is substantially smaller than that of the full flow Jacobian. The linearization of the viscous flux SATs for the Navier-Stokes equations is also modified to ignore the tangential derivatives, which are analogous to the cross-derivatives. Additionally, the viscosity value appearing in the viscous fluxes is treated as a constant when forming the approximate Jacobian.

No approximations are made to the discretization of the turbulence model when constructing the Jacobian entries that arise due to the solution of this extra equation, since all cross-derivatives were dropped during the coordinate transformation and the turbulence model already possesses the minimum stencil size.

The implicit Euler method requires a time step whose inverse is added to the diagonal elements of $\mathcal{A}_1$. A spatially varying time step has been shown to improve the convergence rates of Newton-Krylov algorithms, leading to the use of the following value:

$$\Delta t_{j,k,m}^{(n)} = \frac{J_{j,k,m} \Delta t_{\text{ref}}^{(n)}}{1 + \sqrt[3]{J_{j,k,m}}}, \tag{30}$$

where (j, k, m) denote the indices of the node to which this time step is being applied. Since the solver uses the unscaled flow variables $\mathbf{Q}$, instead of the transformed variables $\hat{\mathbf{Q}}$, the J term that results from the coordinate transformation is lumped into the numerator of (30). The reference time step is

$$\Delta t_{\text{ref}}^{(n)} = a(b)^n,$$

where typical values used for turbulent flow solutions are $a = 0.001$ and $b = 1.3$. The approach provides a steady geometric increase to Δt_{ref}, allowing the algorithm to take progressively larger time steps without destabilizing the solution. This is especially important in the initial iterations, where large fluctuations in flow quantities could lead to divergence.

Effective preconditioning is critical to an efficient parallel linear solver. The first-order Jacobian is factored using block incomplete lower-upper factorization (BILU) with fill level p in order to construct the preconditioner, with a typical fill level of 2. This is a computationally expensive task, especially in the approximate-Newton phase, which requires many outer iterations. Previous work [9, 45] has shown that lagging the update of the preconditioner (freezing it for a number of iterations) during the start-up phase can improve the efficiency of the flow solver. Experience with turbulent flow solutions is mixed, with lagging often resulting in divergence of the nonlinear problem.

Two approaches to parallel preconditioning, namely additive-Schwarz [46] and approximate-Schur [47], have been previously investigated in the context of a parallel Newton-Krylov flow solver for the Euler [9, 24] and Navier-Stokes [24] equations, with a thorough description of their application to the current linear system provided in the references. The approximate-Schur parallel preconditioner is used in the current work, and is described in Section IV C.

An important part of using a start-up phase is knowing when a suitable iterate has been found to initiate the inexact-Newton phase. For this purpose, the relative drop in the residual is used:

$$R_d^{(n)} \equiv \frac{||\boldsymbol{\mathcal{R}}^{(n)}||_2}{||\boldsymbol{\mathcal{R}}^{(0)}||_2}. \tag{31}$$

For turbulent flows, once this value reaches 1×10^{-4}, i.e. the residual has dropped by 4 orders of magnitude in the approximate-Newton phase, the algorithm switches to the inexact-Newton method. This initial drop is larger than is required for inviscid or laminar solutions for two reasons. First, the turbulence quantity fluctuates substantially more than the mean-flow quantities during the start-up phase, necessitating a longer start-up than flow solutions dealing with inviscid or laminar flows. Second, due to the use of grids with much finer spacing near the surface of the aerodynamic shape, the initial residual, $\boldsymbol{\mathcal{R}}^{(0)}$, starts at a much larger value, but drops by one to two orders of magnitude very quickly before settling into a convergence pattern similar to that observed with inviscid or laminar solves. Hence, the relative residual drop threshold, R_d, is adjusted to compensate for these differences. This parameter may need to be adjusted slightly depending on the complexity of the flow being solved.

B. Inexact-Newton phase

The inexact-Newton phase uses a different scheme for the reference time step, designed to ramp the time step toward infinity more rapidly than in the approximate-Newton phase. This eventually eliminates the inverse time term from the left-hand-side of the discretized Navier-Stokes equations. The present work involves the use of a scheme developed by Mulder and van Leer [48], by which a new reference time step is calculated and used in (30):

$$\Delta t_{\text{ref}}^{(n)} = \max\left[\alpha\left(R_d^{(n)}\right)^{-\beta}, \Delta t_{\text{ref}}^{(n-1)}\right],$$

where $\beta \in [1.5, 2.0]$ and α is calculated as

$$\alpha = a(b)^{n_{\text{Newt}}}\left(R_d^{(n_{\text{Newt}})}\right)^{\beta},$$

and n_{Newt} is the first iteration of the inexact-Newton phase.

In contrast with the approximate-Newton phase, the inexact-Newton phase uses the full second-order accurate Jacobian. However, since we use a Krylov subspace method, we do not need to form the full Jacobian matrix, $\mathcal{A}$, explicitly. Instead, only Jacobian-vector products are required, which can be approximated using a first-order forward difference:

$$\mathcal{A}^{(n)}\mathbf{v} \approx \frac{\boldsymbol{\mathcal{R}}(\boldsymbol{\mathcal{Q}}^{(n)} + \epsilon\mathbf{v}) - \boldsymbol{\mathcal{R}}(\boldsymbol{\mathcal{Q}}^{(n)})}{\epsilon}.$$

The parameter ϵ is determined from

$$\epsilon = \sqrt{\frac{N_u \delta}{\mathbf{v}^T\mathbf{v}}},$$

where N_u is the number of unknowns, and $\delta = 10^{-12}$. The approximate Jacobian, $\mathcal{A}_1$, is still used for preconditioning the system.

Finally, neither the approximate-Newton nor the inexact-Newton phase solves its respective linear system exactly. Instead, the following inequality is used to govern how far the system is solved:

$$||\boldsymbol{\mathcal{R}}^{(n)} + \mathcal{A}^{(n)}\Delta\boldsymbol{\mathcal{Q}}^{(n)}||_2 \leq \eta_n ||\boldsymbol{\mathcal{R}}^{(n)}||_2,$$

where the forcing parameter η_n is specified. If it is too small, the linear system will be over-solved and will take too much time, but if it is too large, non-linear convergence will suffer. For the present work, a value of 0.05 is used for the approximate-Newton phase, while 0.01 is used for the inexact-Newton phase. An important aspect to consider when using GMRES is the amount of memory available on the computational hardware. Since GMRES stores the vectors from the individual search directions, it is important to limit the maximum number that are stored (to prevent the algorithm from crashing). This is especially critical in the inexact-Newton phase, where the linear solver may have difficulty attaining the above-mentioned relative tolerance before the maximum number of search directions is reached, especially for complex flows. For the flow solutions presented in this work, limiting the number of search directions to 70 provides sufficient linear convergence without reaching memory limitations of the computational systems. It should be stressed that this limit is typically triggered towards the end of the flow solution process.

C. Approximate-Schur Parallel Preconditioner

The preconditioning step requires the solution of the generic global system

$$\mathsf{A}\mathbf{x} = \mathbf{b}, \tag{32}$$

where $\mathsf{A} \in \mathbb{R}^{n\times n}$ and $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$.

In the context of a distributed parallel implementation, it is necessary to define a representation of the individual subsystems located on various processes. To achieve this, the rectangular matrix P_i is used to project the global vector onto a local vector corresponding to the unknowns stored on process i. Applying P_i to the linear system Eq. (32), and considering local solutions of the form $\mathbf{x} = \mathsf{P}_i^T \mathbf{x}_i$, we obtain the block diagonal system

$$\mathsf{A}_i \mathbf{x}_i = \mathbf{b}_i, \tag{33}$$

where $\mathbf{b}_i \equiv \mathsf{P}_i \mathbf{b}$ and $\mathsf{A}_i \equiv \mathsf{P}_i \mathsf{A} \mathsf{P}_i^T$. The local matrix A_i is factored into $\mathsf{L}_i\mathsf{U}_i$ using BILU(p) [49].

The local unknowns, $\mathbf{x}_i$, are separated into two groupings, consisting of unknowns dependent on local information only, $\mathbf{u}_i$, and unknowns that are coupled to information on processor $j \neq i$, defined as $\mathbf{y}_i$. The coupling is a result of the application of SATs at block interfaces. By placing the unknowns $\mathbf{y}_i$ last, this ordering partitions $\mathsf{P}_i\mathsf{A}\mathbf{x} = \mathsf{P}_i\mathbf{b}$ into the following block structure:

$$\begin{pmatrix} \mathsf{B}_i & \mathsf{F}_i \\ \mathsf{H}_i & \mathsf{C}_i \end{pmatrix} \begin{pmatrix} \mathbf{u}_i \\ \mathbf{y}_i \end{pmatrix} + \begin{pmatrix} 0 \\ \sum_j \mathsf{E}_{ij}\mathbf{y}_j \end{pmatrix} = \begin{pmatrix} \mathbf{f}_i \\ \mathbf{g}_i \end{pmatrix}. \tag{34}$$

The variables $\mathbf{u}_i$ are not coupled across processors and can be eliminated using

$$\mathbf{u}_i = \mathsf{B}_i^{-1}(\mathbf{f}_i - \mathsf{F}_i \mathbf{y}_i). \tag{35}$$

Substituting $\mathbf{u}_i$ into Eq. (34) we find the following system for the variables coupling the domains:

$$\underbrace{\begin{pmatrix} \mathsf{S}_1 & \mathsf{E}_{12} & \dots & \mathsf{E}_{1P} \\ \mathsf{E}_{21} & \mathsf{S}_2 & \dots & \mathsf{E}_{2P} \\ \vdots & & \ddots & \vdots \\ \mathsf{E}_{P1} & \mathsf{E}_{P2} & \dots & \mathsf{S}_P \end{pmatrix}}_{\mathsf{S}} \begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_P \end{pmatrix} = \begin{pmatrix} \mathbf{g}'_1 \\ \mathbf{g}'_2 \\ \vdots \\ \mathbf{g}'_P \end{pmatrix}, \tag{36}$$

where $\mathsf{S}_i \equiv \mathsf{C}_i - \mathsf{H}_i\mathsf{B}_i^{-1}\mathsf{F}_i$, and $\mathbf{g}'_i \equiv \mathbf{g}_i - \mathsf{H}_i\mathsf{B}_i^{-1}\mathbf{f}_i$. The coefficient matrix S is the Schur complement corresponding to the variables coupled between processors. Suppose we solve Eq. (36) using block Jacobi. This approach would parallelize well, but it requires S_i — more precisely, its inverse — which can be expensive to form explicitly. Saad and Sosonkina [47] recognized that an ILU factorization of S_i can easily be extracted from an ILU(p) factorization of A_i. Their Schur-based preconditioner consists of a GMRES-accelerated approximate solution of Eq. (36), with S_i replaced by its ILU factorization. Once approximate solutions to the $\mathbf{y}_i$ are obtained, they are substituted into Eq. (35), with B_i replaced with its ILU(p) factorization, to obtain $\mathbf{u}_i$.

D. Special Considerations for Turbulence Model

1. *Equation scaling*

The addition of the turbulence model to the linear system Eq. (29) presents some unique challenges, as the scaling of the linear system can be adversely affected, resulting in unpredictable behavior of the linear solver. The improper scaling arises from several factors. First, the turbulence model does not contain the inherent geometric scaling present in the mean flow equations (division by J). Second, the turbulence quantity can be as large as 1000 or higher in the converged solution, while the nondimensionalized mean flow quantities rarely exceed 2. Finally, the terms that result from the linearization of the turbulence model with respect to the mean flow variables add large off-diagonal values to the Jacobian. Hence, a more sophisticated scaling approach has been implemented to account for these discrepancies, based on Chisholm and Zingg [7], in order to obtain an efficient and accurate solution of the linear system. The row, or equation, scaling of the mean flow equations is achieved by multiplying the equations by a factor that includes the metric Jacobian, removing the inherent geometric scaling, while the turbulence model is scaled by $\tilde{\nu}_{\max}^{-1}$. This value accounts for the maximum turbulence value that is likely to be encountered in the flow solve, effectively normalizing the turbulence equation by that quantity. For the current work, $\tilde{\nu}_{\max} = 10^3$. In order to normalize the flow variable values, the turbulence variable quantity is also multiplied by $\tilde{\nu}_{\max}^{-1}$. Hence, instead of solving the system presented in Eq. (29), the linear solution algorithm tackles a scaled system of the form

$$S_{\mathrm{a}} S_{\mathrm{r}} \left(\frac{\mathcal{I}}{\Delta t} + \mathcal{A}^{(n)} \right) S_{\mathrm{c}} S_{\mathrm{c}}^{-1} \Delta \boldsymbol{\mathcal{Q}}^{(n)} = -S_{\mathrm{a}} S_{\mathrm{r}} \boldsymbol{\mathcal{R}}^{(n)}, \tag{37}$$

where S_{r} and S_{c} are the row and column scaling matrices, respectively. S_a is an auto-scaling matrix used to bring the values of the individual equation components within an order of magnitude, further improving the scaling of the linear system. In the current implementation, these matrices are defined as

$$S_{\mathrm{r}} = \mathrm{diag}\left(S_{\mathrm{r}1}, ..., S_{\mathrm{r}N}\right), \quad S_{\mathrm{c}} = \mathrm{diag}\left(S_{\mathrm{c}1}, ..., S_{\mathrm{c}N}\right),$$

where

$$S_{\mathrm{r}i} = \begin{bmatrix} J_i^{2/3} & & & & & \\ & J_i^{2/3} & & & & \\ & & J_i^{2/3} & & & \\ & & & J_i^{2/3} & & \\ & & & & J_i^{2/3} & \\ & & & & & \tilde{\nu}_{\max}^{-1} J_i^{-1/3} \end{bmatrix}, \quad S_{\mathrm{c}i} = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & & 1 & \\ & & & & & \tilde{\nu}_{\max} \end{bmatrix},$$

and J_i is the value of the metric Jacobian at the i^{th} node in the computational domain. The powers used for the metric Jacobian are selected to reduce the dependency on the mesh spacing in the residual. The values in the auto-scaling matrix are calculated based on the equation-wise residual L_2-norms of the partially scaled system $S_{\mathrm{r}} \boldsymbol{\mathcal{R}}^{(n)}$. Instead of scaling each node by a unique value, they scale the component equations by different amounts. Any residual values required for tracking convergence and the time step calculation make use of the partially scaled residual $S_{\mathrm{r}} \boldsymbol{\mathcal{R}}^{(n)}$, which balances the contributions from the component equations, particularly the turbulence model.

2. Negative turbulence quantities

During convergence to steady state, it is not atypical to encounter negative values of $\tilde{\nu}$ in the flowfield. These values are nonphysical, and merely a result of the occurrence of large transients in the solution, especially during the early stages of convergence or after the switch from the approximate-Newton phase to the inexact-Newton phase. However, it is important to address these negative values, since they could destabilize the solution process. The approach taken is to trim any negative $\tilde{\nu}$ values to a very small positive quantity. In particular, any negative turbulence quantities that are encountered on a solid surface are trimmed to $10^{-14}\nu$, while all other locations are trimmed to $10^{-3}\nu$. The local ν value is introduced such that advective and diffusive fluxes do not vanish completely from regions where several adjacent nodes are trimmed during the same iteration, which can occur if all values are trimmed to a constant.

Additional trimming is used when dealing with the value of vorticity, S. In order to avoid numerical problems, this value is not allowed to fall below 8.5×10^{-10}. Finally, the vorticity-like term, $\tilde{S}$, cannot be allowed to become nonpositive, which would have a destabilizing effect on the values of the production and destruction terms. We have found that the following trimming approach works well:

$$\tilde{S} = \begin{cases} \tilde{S} & : \ \tilde{S} > 0.3S \\ 0.3S & : \ \text{otherwise.} \end{cases} \tag{38}$$

3. Off-wall distance calculation

An important consideration in the use of the Spalart-Allmaras turbulence model is the method by which the off-wall distance, d, is calculated. Many implementations, in an effort to save computational time and reduce code complexity, make use of approximations in calculating the distance from each volume node to the closest solid boundary. These range from following grid lines towards a surface, a method that itself becomes very complex when dealing with multi-block grids, to calculating distances to individual grid nodes on the surface. While these methods provide reasonably accurate representations of d, certain grid topologies can result in values that have a substantial effect on the force coefficients. This effect is due to the sensitivity of the Spalart-Allmaras turbulence model to the values used for d. It is therefore highly recommended that a method which identifies the entire surface, including edges and surface segments between the nodes on a solid boundary, be used in calculating d, as it will provide more accurate off-wall distance values on a wide variety of grids, regardless of blocking or gridding anomalies. Such an approach is used in the current algorithm.

4. Time step for turbulence model

In their work on a two-dimensional Newton-Krylov flow solution algorithm, Chisholm and Zingg [7] discuss the need to introduce limits on the time step used for the turbulence model. In their experience, the limits were required to achieve robust performance of the algorithm, which could destabilize if large updates to the turbulence variable were made in the early stages of convergence.

For a strong boundary condition treatment, the large fluctuations in the turbulence quantities are usually observed near solid boundaries and are the result of a large difference between the initial values of the turbulence quantities of the surface and the surface-adjacent nodes. Through the use of a strong boundary condition, the surface values are forced to remain at a constant value throughout the solution process, regardless of what is happening elsewhere in the flowfield. This can result in large flux values, eventually leading to large, destabilizing solution updates. By limiting the time step used for the turbulence model, the large updates to the turbulence variable are significantly reduced, resulting in more robust performance over a range of flow conditions. The SBP-SAT discretization, on the other hand, does not initially differentiate between surface and surface-adjacent nodes; all nodes are initialized to the same flow variable values. Instead, the SATs act to weakly enforce the boundary conditions, slowly changing the values of the surface variables as the solution evolves. In this way, the SBP-SAT discretization rarely sees large updates to the turbulence quantity, and therefore does not necessitate the use of special time step values for the turbulence model when solving fully turbulent flows. Similar convergence improvements have been demonstrated by Nordström *et al.* [50], who have shown that weakly enforced boundary conditions provide faster convergence to steady state for the Navier-Stokes equations.

In fact, the only special time step limitations required for the turbulence model used in this algorithm are related to explicitly tripped turbulent flow solutions. As expected, the substantial nonlinearities present in such a flow pose a difficult problem for the solution algorithm, necessitating the use of a reduced time step. This was also true of the work done in [7], and the current approach is based on the results presented therein.

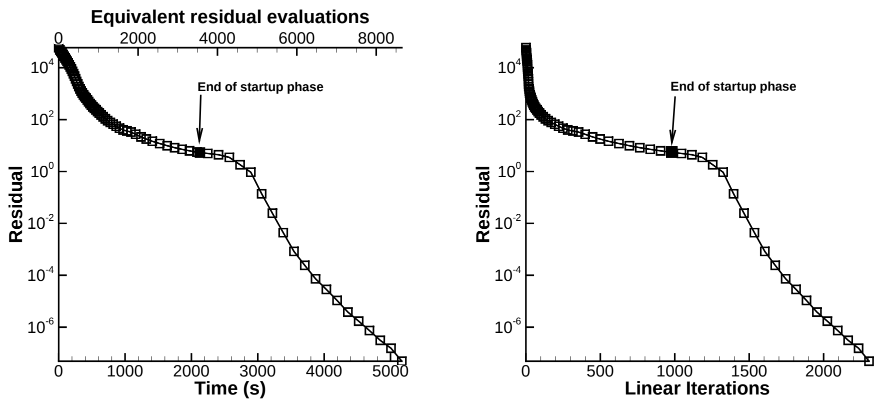
The use of the explicit trip terms in the turbulence model, when simulating flows with laminar to turbulent transition, can introduce substantial instabilities into the solution process, often resulting in divergence. An effective strategy is to limit the time step used for the turbulence model, while the mean-flow equation time step remains unchanged. In particular, the following has been found to work well for cases when explicit trip terms are active:

$$\Delta t_{\mathrm{SA\_trip}} = \frac{1}{100}\Delta t, \tag{39}$$

where Δt is is defined in Eq. (30). This modification is not necessary for fully turbulent flow simulations.

V. Results

This section presents a number of cases that were run to showcase the capabilities of the flow solver on a wide range of challenging steady flow computations. Included are solutions of flow



a) Vs. time   b) Vs. linear iterations

Fig. 3 Residual convergence for transonic flow over the ONERA M6 wing with 128 processors

around wing and wing-body configurations, wings with explicitly specified laminar-to-turbulent transition locations, and non-planar configurations. Finally, the parallel scaling of the algorithm is demonstrated on very large grids, using up to 6656 processors.

In order to verify the implementation and accuracy of the current algorithm, several verification and validation cases were completed. Making use of the extensive data available on the NASA Turbulence Modeling Resource (TMR) website (`http://turbmodels.larc.nasa.gov`), we have been able to verify the current algorithm, named DIABLO, against well-established solvers FUN3D and CFL3D for the flat plate and bump-in-channel flows. Additionally, validation studies were performed with flow around a NACA0012 airfoil, where comparison to experimental results could also be made. These results can be found in the Appendix.

Computations were performed on the GPC supercomputer at the SciNet HPC Consortium and the Guillimin supercomputer of the CLUMEQ consortium, both part of Compute Canada. The systems use Intel Nehalem processors interconnected with non-blocking 4x- and 8x-DDR InfiniBand.

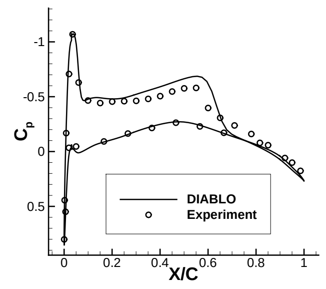
A. ONERA M6 Wing

The first case considered is the well known transonic flow over the ONERA M6 wing, for which the experimental data of Schmitt and Charpin are available [51]. The flow conditions are

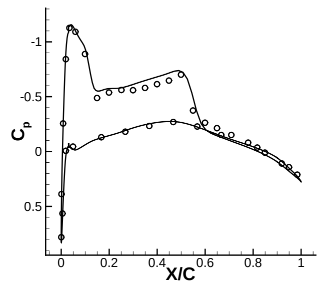
$$M = 0.8395,\ Re = 11.72 \times 10^6,\ \alpha = 3.06^\circ.$$

The Reynolds number is based on the mean aerodynamic chord (MAC) of the geometry. The grid used in this study consists of 128 blocks, with a total of 15.1 million nodes and an off-wall spacing of 2.3×10^{-7} root chord units, resulting in an average y^+ value of 0.4. The flow solution was computed using the scalar dissipation model. With 128 processors, the residual was reduced by 12 orders of magnitude in 86 minutes. Figure 3 presents the residual convergence plot for this case, both in terms of time and linear iterations, clearly highlighting the two phases of the solver. The residual includes all six equations. Convergence is also plotted versus “equivalent residual evaluations,” normalizing the solution time by the time required to calculate the residual for the size of grid being used in the flow solution. Since implicit algorithms spend substantial time in forming and factoring the preconditioner, as well as solving the linear system at each nonlinear iteration, this time measure provides a clearer comparison to explicit solution algorithms, for which residual evaluations dominate the cost of nonlinear iterations. The switch from the approximate-Newton phase to the inexact-Newton phase occurs at approximately 35 minutes. This plot represents a typical convergence history for DIABLO, with the symbols along the line showing individual nonlinear iterations. The simulation results in values of C_L and C_D of 0.268 and 0.0171, respectively.

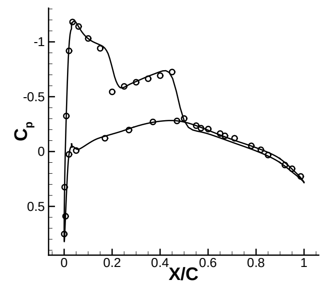
Figure 4 presents the C_p distributions at several span-wise sections of the wing, comparing to experimental data. The comparison highlights the good correspondence of the computational and experimental results throughout the span of the wing. This case demonstrates the capability



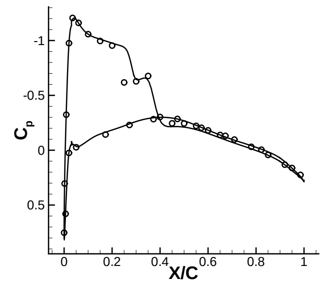
a) 20% span



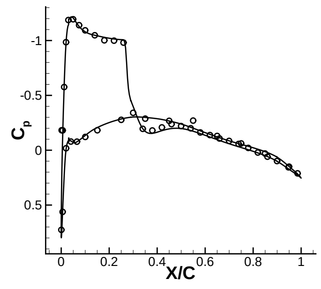
b) 44% span



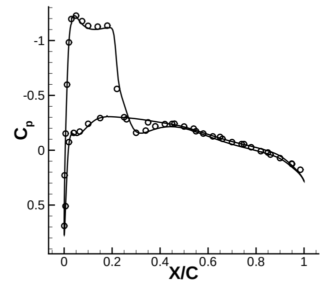
c) 65% span



d) 80% span



e) 90% span



f) 95% span

Fig. 4 Experimental and numerical C_p distributions for ONERA M6 wing flow

of the current algorithm to accurately and efficiently capture the complex transonic flow around the ONERA M6 wing. In order to determine the cause of the small discrepancies between the numerical and experimental results, the solution was computed on a finer, 128 million node grid, with an average y^+ value of 0.18. The values of C_L and C_D changed slightly to 0.269 and 0.0168, respectively, but the C_p distributions remain virtually identical to those produced on the 15.1 million node grid. The observed C_p distribution differences may be caused by differences in the problem set-up between the numerical and experimental cases, such as the use of free transition in the experimental measurements, as opposed to the assumption of fully turbulent flow in the present computations.

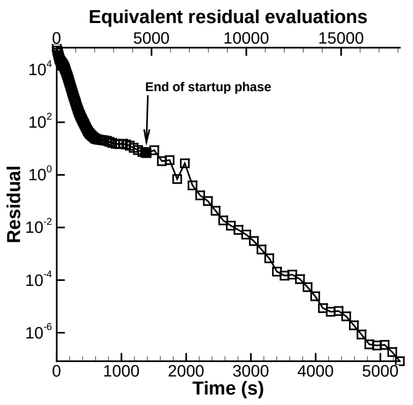
B. NASA Common Research Model Wing-Body Configuration

This case is a transonic flow around the CRM wing-body geometry. The O-O topology structured multi-block grids, obtained from the organizers of the 5th Drag Prediction Workshop (DPW5), contain between 750 thousand and 154 million nodes, with an off-wall spacing range of 2.4×10^{-6} to 1.9×10^{-7} MAC units, corresponding to y^+ values of 2.0 to 0.33. The original grid family was constructed using a 2-to-3-to-4 node refinement strategy, detailed by Vassberg [52]. The six grid levels effectively comprise two nested grid families, with the odd and even grid levels following the typical 1-to-2 node refinement strategy. The grids contain between 88 and 906 blocks, depending on the size of the grid, but were not load balanced perfectly due to the original grid construction. The flows were computed at flow conditions of

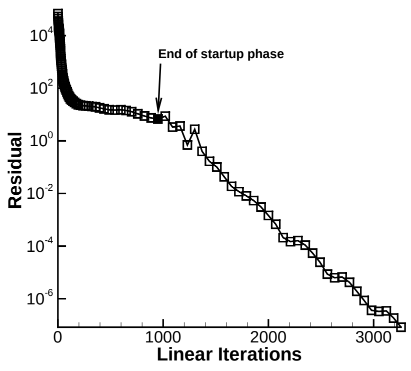
$$M = 0.85,\ Re = 5 \times 10^6,\ C_L = 0.500 \pm 0.001.$$

The Reynolds number is based on the MAC. Each grid level requires a different angle of attack to attain the specified value of C_L. Both scalar and matrix dissipation are examined.

Solutions were obtained on six grid levels, converging the residual by 10 orders of magnitude on each grid. Figure 5 presents the convergence history for the 19 million node fine ("F") grid level with matrix dissipation. This particular solution was computed on 704 processors, which were able to converge the solution in 70 minutes. A load balancing approach, as detailed by Apponsah and



a) Vs. time



b) Vs. linear iterations

Fig. 5 Convergence history for CRM flow on 19 million node grid (with 704 processors)

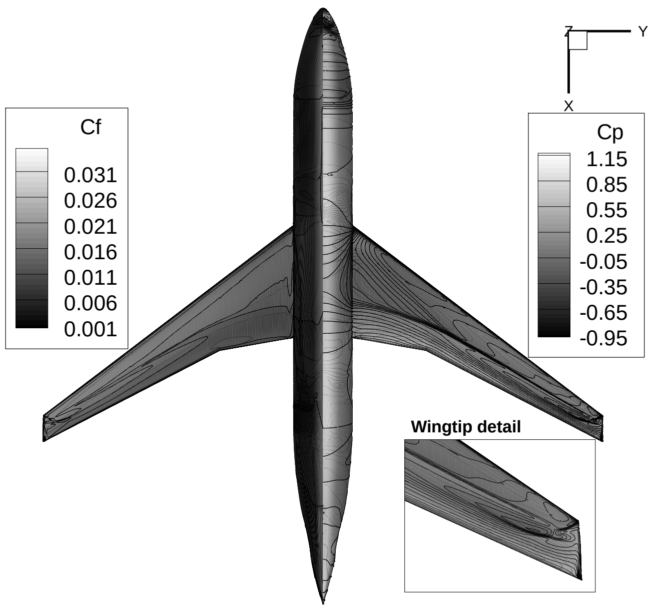
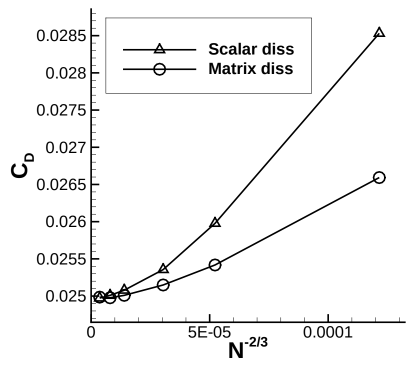


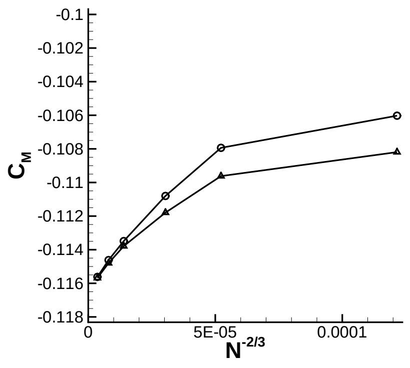
Fig. 6 Surface coefficients for CRM flow

Zingg [53], can be used to reduce the time required substantially. As with the previously presented ONERA M6 test case, both stages of the solution algorithm are clearly visible. However, due to the complexity of the flow, more iterations are required in the inexact-Newton stage. Figure 6 shows the contours of C_p and C_f on the top surface of the geometry. The pressure contours highlight the presence of a shock on the top surface of the wing, with a complex interplay of two shocks near the wingtip.

This case also allows us to observe the grid convergence of the algorithm. Figure 7 presents the grid convergence trends of both drag and moment coefficients, with the matrix dissipation model producing a flatter curve for drag convergence. This signifies that, as expected, the model is more accurate on coarser grids. Additionally, a second-order algorithm will tend toward a straight line between three consecutive grid levels when plotted versus $N^{-2/3}$. This behavior can be observed for the finer grid levels. Using Richardson extrapolation [54] on the three finest grid levels, the order of convergence for C_D is calculated to be 1.60 and 3.32 for the scalar and matrix artificial dissipation models, respectively, exhibiting the expected, or better, grid convergence characteristics. These convergence rates are achieved despite the first-order treatment of the convective terms in the turbulence model, indicating that these terms do not dominate the discretization error and appear

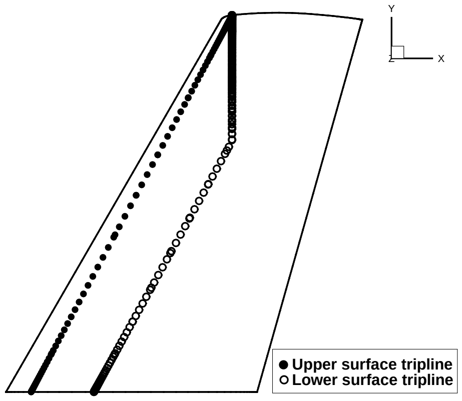


a) Drag coefficient

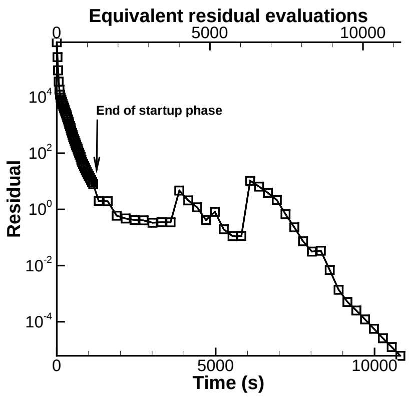


b) Pitching moment coefficient

Fig. 7 Grid convergence of drag and pitching moment coefficient for CRM flow



a) Wing planform and trip line definition



b) Convergence history

Fig. 8 Explicitly tripped flow with ONERA M6 wing

to play a small role. Both dissipation models tend towards the same result as the grid is refined.

C. Explicitly tripped flow solutions

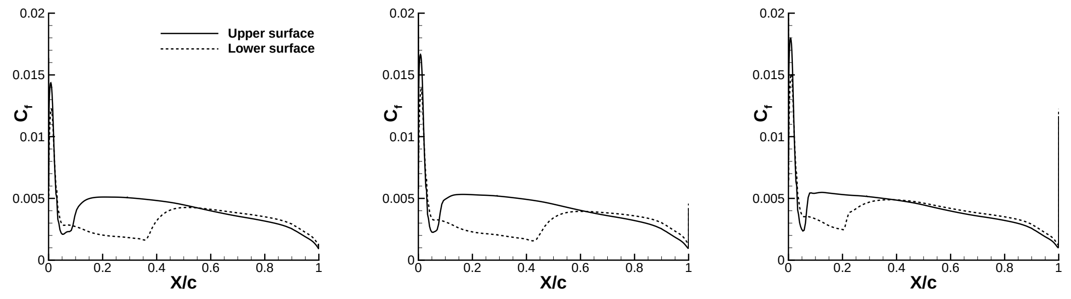
Further extending the capability of the algorithm, we have implemented the explicit trip location terms of the Spalart-Allmaras turbulence model. These terms require the user to specify an explicit location for laminar to turbulent transition to occur. Eventually, the ability to specify the transition location will be coupled with transition prediction, such that the flow solver will be incorporated into an optimization algorithm that can take advantage of natural laminar flow in improving the drag characteristics of an aerodynamic shape.

The case considered involves the ONERA M6 wing. The flow conditions for this case are

$$M = 0.30,\ Re = 1 \times 10^6,\ \alpha = 1.00^\circ.$$

The grid used is identical to that in Section V A. The transition line is specified individually on the upper and lower surfaces of the wing, as shown in Fig. 8a). Only the scalar dissipation model was used for this case.

The addition of the explicit trip terms increases the initial magnitude of the residual, so it is necessary to remain in the start-up phase for one additional order of magnitude until the residual drops by a factor of 1×10^{-5}. It is for this case that the time step modification discussed in Section IV D 4 is critical; without it, the solution would diverge due to large changes in the turbulence quantity. Additionally, the complexity of the flow causes noticeable jumps in the residual, which



a) 20% span b) 65% span c) 90% span

Fig. 9 C_f values at selected span-wise sections for tripped flow

results in longer convergence times than for fully turbulent solutions. This is likely due to the locations of the trip lines themselves, forcing the flow to remain laminar over large portions of the wing, especially on the lower surface. Other than these jumps, the convergence history in Fig. 8b) shows the typical convergence characteristics of the Newton-Krylov algorithm.

Additionally, the C_f plots at selected span-wise sections, shown in Fig. 9, demonstrate the effect of imposing specific trip locations. The C_f values along both surfaces show the characteristic increase at the specified location of laminar to turbulent transition.

D. Non-planar geometry

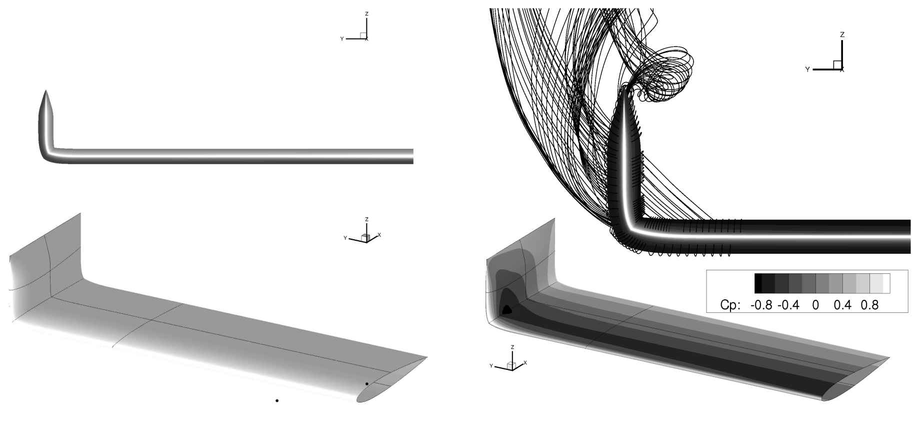
With computational analysis and optimization becoming more integrated in the design process of modern aircraft, flow solution algorithms have to be capable of handling more complex and unconventional geometries. For example, non-planar geometries are becoming commonplace and require computational analysis to fully understand their trade-offs relative to conventional wings. We have found these cases to possess unique flow features which may introduce numerical instabilities and hamper a solver's convergence to steady state. The test case considered here involves an unswept rectangular wing with a vertical winglet, as shown in Fig. 10a). Both the wing and winglet possess the NACA 0012 airfoil cross-section. The NACA0012 airfoil geometry is extruded (in ICEM CFD) along a line that defines the leading edge of the wing, with tapering applied to close the vertical tip. The flow conditions are

$$M = 0.40,\ Re = 7.48 \times 10^6,\ \alpha = 2.00^\circ.$$

The finest computational grid consists of 960 blocks, with a total of 113 million nodes and an off-wall distance of 4.0×10^{-7} chord units, giving an average y^+ of 0.18. Two additional grid levels were created by successively removing every second node in each computational direction. On the finest grid, the solution was obtained using 960 processors, converging the residual by 12 orders of magnitude in 125 minutes, as shown in Fig. 11. The results for this case illustrate the convergence capabilities of the current algorithm when dealing with non-planar geometries. Only the scalar dissipation model was used for this case. Figure 10b) shows the C_p contours on the surface of the geometry, as well as some streamlines emanating from the winglet area.

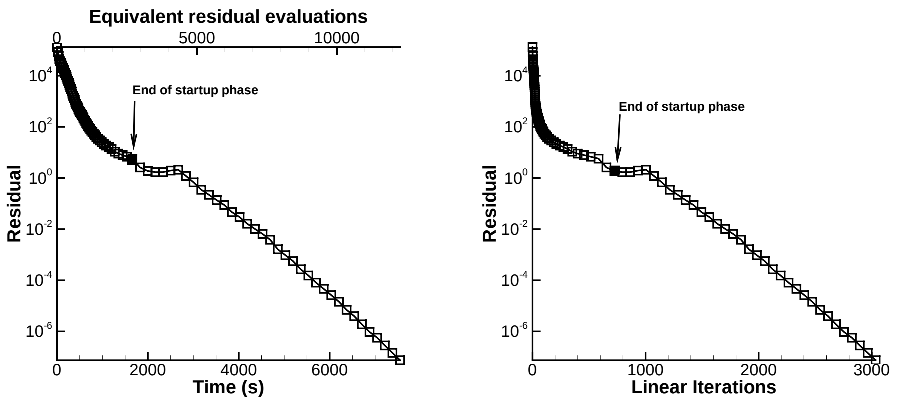
The nested grid family allows us to use Richardson extrapolation to compute the order of convergence of lift and drag. Table 1 presets the force coefficients on all grid levels, in addition to the calculated orders of convergence, p, and grid-converged force values, F^*. The orders of convergence for C_L and C_D are 1.80 and 2.32, respectively, exhibiting the expected grid convergence characteristics for the second-order spatial discretization. The planar area of 3.041 is used as the reference area.

Non-planar geometries provide challenges in terms of grid generation compared to conventional wings, especially in the area of the winglet. The use of a multi-block gridding strategy allows us to place blocks and refine the grid around the geometry in a manner that resolves the boundary layer of both the wing and winglet without introducing excessive grid stretching. Additionally, the SBP-SAT



a) Winglet geometry b) Steamlines and C_p contours

Fig. 10 Non-planar geometry and flow



a) Vs. time b) Vs. linear iterations

Fig. 11 Non-planar wing convergence history on 133 million node grid (with 960 processors)

discretization used in the current algorithm provides excellent flexibility in terms of the blocking strategy used, further extending the ease with which complex geometries can be accommodated.

E. Parallel scaling of algorithm

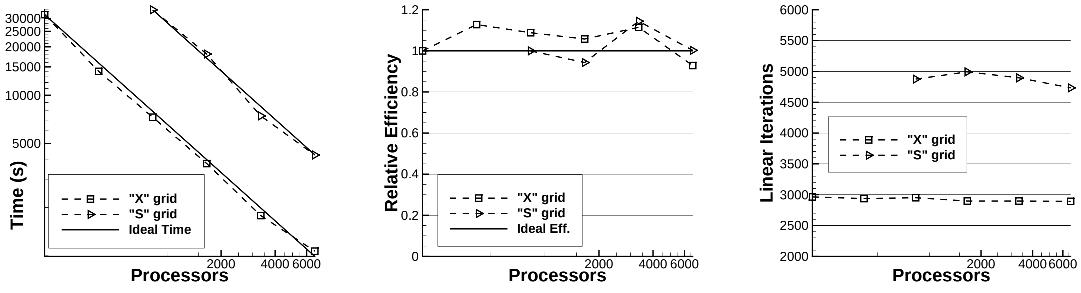
In order to evaluate the parallel performance of the algorithm, a parallel scaling study was conducted for the solution of a transonic flow around the CRM wing-body geometry. The flow conditions are identical to those in Section V B.

The CRM wing-body case was run on two grids, with each grid consisting of a 6656-block O-O topology mesh obtained from the "X" and "S" grid levels used in DPW5. The original 5-block grids are sequentially subdivided until all blocks contain the same number of nodes, resulting in the final 6656-block grids. The "X" grid contains 48 million nodes and an off-wall spacing of 1.83×10^{-6} MAC units, while the "S" grid contains 154 million nodes and an off-wall spacing of 1.29×10^{-6} MAC units. The matrix dissipation model is used, and the residual is converged 10 orders of magnitude.

The results for both cases, presented in Fig. 12, show that the code exhibits excellent parallel scaling characteristics up to 6656 processors. All processor levels used the same grid and the same number of blocks, with the results providing a measure of the strong scaling characteristics of the algorithm. The performance of the code is measured by relative efficiency, which is based on the lowest possible number of processors that each case can be computed with. Due to memory requirements, the cases require a minimum of 208 and 832 processors, respectively. The nearly constant number of linear iterations required to converge the solutions at different processor numbers

Table 1 Grid convergence for C_L and C_D

Grid level	Grid nodes	C_L	C_D
coarse	2,109,120	0.1773	0.01676
medium	15,000,000	0.1820	0.01187
fine	112,943,040	0.1834	0.01087
order, p	-	1.80	2.32
F^*	-	0.1840	0.01064



a) Time b) Relative parallel efficiency c) Linear iterations

Fig. 12 Parallel scaling performance of DIABLO

highlights the effectiveness of the approximate-Schur preconditioner even when large numbers of processors (>6000) are used. In the range of processors considered, the relative efficiency does not drop below 80%. In fact, many processor counts exhibit super-linear scaling, possibly due to the changing form of the preconditioner, with different numbers of interface nodes contributing to the global Schur complement. The size of the grid, and the corresponding memory requirements, places limitations on the number of processors available on each compute node (with a maximum of eight processors per node); this impacts the parallel performance of the algorithm. For example, the baseline cases require the use of one processor per compute node, resulting in all parallel communication occurring between nodes (inter-node), whereas the larger processor counts can make use of several processors per node, allowing for intra-node communication. Due to the inherent characteristics of the system design, inter-node communication is slower than intra-node communication, resulting in the lower processor count cases possessing a higher effective communication overhead. No effort was made to optimize the communication overhead for any of the cases through assigning processors to specific compute nodes to minimize inter-node communication.

The scaling characteristics of the code, coupled with the ease with which the SBP-SAT approach can handle arbitrary numbers of blocks and their orientations, make this algorithm an attractive option for applications where fast turnaround times are required. Not only is the underlying parallel Newton-Krylov-Schur algorithm robust and efficient in obtaining converged steady-state solutions, it can easily make use of larger numbers of processors, when available, to further reduce wall-time required for computations without a significant loss of computational resource efficiency.

VI. Conclusions

A parallel Newton-Krylov-Schur flow solution algorithm with a second-order SBP-SAT finite-difference spatial discretization was presented and shown to efficiently provide accurate numerical solutions to the three-dimensional Reynolds-averaged Navier-Stokes equations with the one-equation Spalart-Allmaras turbulence model. Modifications to the scaling of the linear system, as well as the time step used for the turbulence model when solving explicitly tripped flows, allow the solver to converge well a wide range of flows. These include subsonic and transonic flows over planar and non-planar aerodynamic geometries, with attached and separated boundary layers.

The algorithm was verified with a selection of two-dimensional test problems, comparing well against established flow solution algorithms, while the solution of transonic flow over the ONERA M6 wing compares well to experimental data. The grid convergence characteristics of the algorithm were demonstrated with the solution of a transonic flow over the Common Research Model (CRM) wing-body geometry on grids with up to 150 million nodes. Additional solutions of explicitly tripped flows and flows over non-planar geometries further highlight the versatility of the current flow solver, demonstrating its suitability for use as the core of an aerodynamic shape optimization capability or a high-fidelity multidisciplinary optimization capability. Good parallel efficiency of the algorithm was demonstrated through the solution of transonic flow over the CRM wing-body geometry with up to 6656 processors.

Future work will extend the spatial discretization of the governing equations to higher-order, with the goal of further improving the efficiency of the algorithm.

Appendix

A. $B_{\text{int},\varsigma}$ matrix formulation

The following are the complete forms of the conservative formulation of the $B_{\text{int},\varsigma}$ matrices, where $\varsigma = \xi, \eta,$ or ζ.

$$B_{\text{int},\varsigma} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -a_1u - a_2v - a_3w & a_1 & a_2 & a_3 & 0 \\ -a_2u - a_4v - a_5w & a_2 & a_4 & a_5 & 0 \\ -a_3u - a_5v - a_6w & a_3 & a_5 & a_6 & 0 \\ b_{51} & b_{52} & b_{53} & b_{54} & a_7 \end{bmatrix},$$

where

$$\begin{aligned}
a_1 &= t_1\left({}^4\!/\!{}_3\varsigma_x^2 + \varsigma_y^2 + \varsigma_z^2\right), & a_2 &= t_1\left({}^1\!/\!{}_3\varsigma_x\varsigma_y\right), \\
a_3 &= t_1\left({}^1\!/\!{}_3\varsigma_x\varsigma_z\right), & a_4 &= t_1\left(\varsigma_x^2 + {}^4\!/\!{}_3\varsigma_y^2 + \varsigma_z^2\right), \\
a_5 &= t_1\left({}^1\!/\!{}_3\varsigma_y\varsigma_z\right), & a_6 &= t_1\left(\varsigma_x^2 + \varsigma_y^2 + {}^4\!/\!{}_3\varsigma_z^2\right), \\
a_7 &= t_2\gamma\left(\varsigma_x^2 + \varsigma_y^2 + \varsigma_z^2\right), & & \\
b_{52} &= -a_7u + a_1u + a_2v + a_3w, & b_{53} &= -a_7u + a_2u + a_4v + a_5w, \\
b_{54} &= -a_7u + a_3u + a_5v + a_6w, & & \\
t_1 &= \rho^{-1}\left(\mu + \mu_t\right), & t_2 &= \rho^{-1}\left(\mu/Pr + \mu_t/Pr_t\right),
\end{aligned}$$

and

$$b_{51} = a_7\left(-e/\rho + (u^2 + v^2 + w^2)\right) - a_1u^2 - a_4v^2 - a_6w^2 - 2(a_2uv + a_3uw + a_5vw).$$

B. Verification and validation of flow solver

2D zero-pressure-gradient flat plate

The first verification case considered was the two-dimensional flow over a flat plate. The flow conditions for this case are

$$M = 0.20,\ Re = 5 \times 10^6,\ T_{\text{ref}} = 540°\text{R}.$$

Three grid levels are considered, with node numbers ranging from 137×97 to 545×385. Successively finer off-wall spacing values are used, the finest of which is 5.0×10^{-7} chord units (the flat plat has a length of 2.0 chord units). The coarser meshes were created by successively removing every second node in each coordinate direction from their respective finer counterpart. The grids provide average approximate y^+ values between 0.1 and 0.4, depending on the grid level. The data provided on the TMR website allows for a detailed comparison of the results obtained with the current algorithm with those obtained with CFL3D and FUN3D. Both scalar and matrix dissipation models were tested with this case.

Figure 13 shows the behavior of drag as the grid is refined, where N denotes the number of nodes in the grid. As a result of plotting versus $N^{-1/2}$, second-order behaviour should tend toward

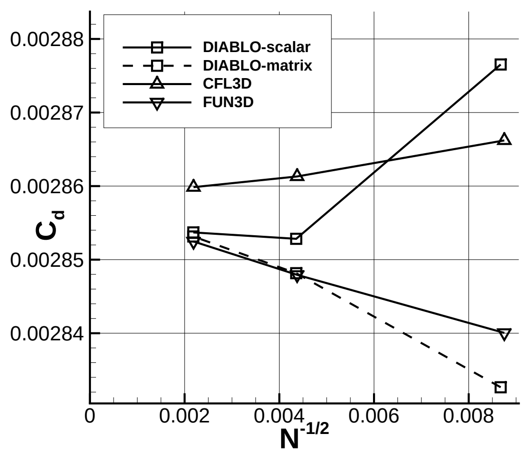


Fig. 13 Grid convergence of drag for flow over a flat plate

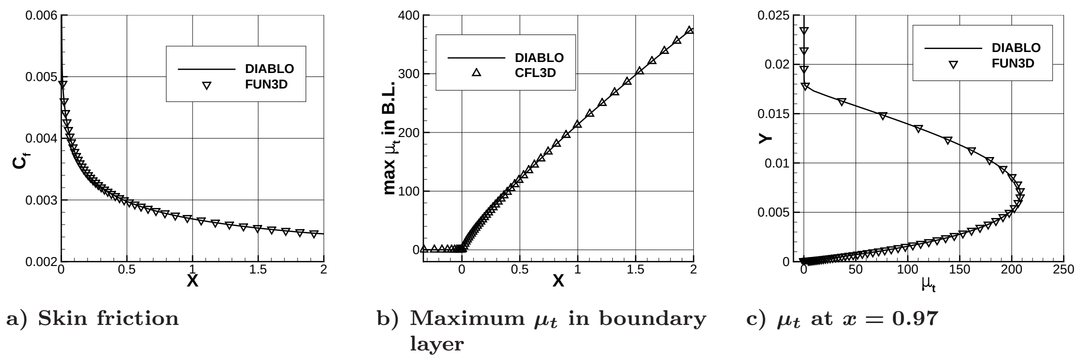


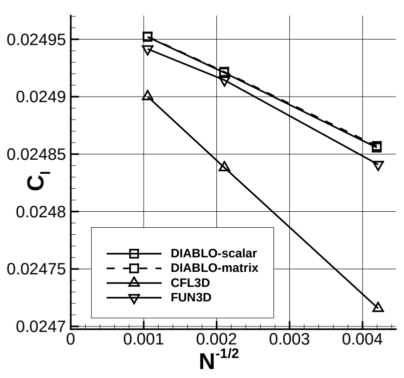
Fig. 14 Flow solution comparisons for flow over flat plate

a straight line. Both dissipation models tend toward the same value of C_d as the grids are refined. However, the scalar dissipation model approaches this value in a non-monotonic manner from above, while the matrix dissipation model closely follows the trend of FUN3D, approaching from below. Both models are tending towards a grid-converged value of drag that lies between the trends of CFL3D and FUN3D.

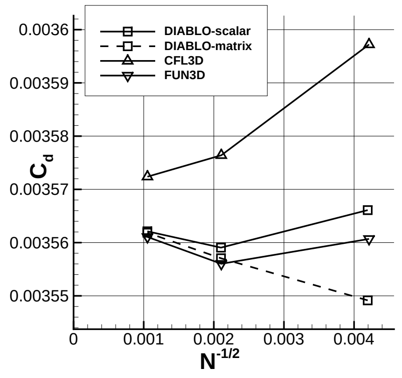
Comparisons of the coefficient of skin friction, C_f, maximum turbulent viscosity in the boundary layer, and the turbulent viscosity profile at a vertical section of the finest grid are presented in Fig. 14. Since the scalar and matrix dissipation results on this grid level are nearly indistinguishable, only the matrix dissipation result is shown. Some data points are omitted for clarity. Each of the comparisons shows excellent correspondence between DIABLO and the other algorithms, with nearly identical distributions of all pertinent quantities.

2D bump-in-channel

In order to verify the algorithm in a more complex flow regime where pressure gradients are present, the second case considered was the two-dimensional bump-in-channel flow. Three grid levels are considered, with node numbers ranging from 353×161 to 1409×641, with successively finer off-wall spacing values, the finest of which is 5.0×10^{-7} chord units (the bump has a length of 1.5 chord units). As with the previous case, the coarser grid levels were created by removing every second node in each coordinate direction from the finer grid level. The grids provide average approximate y^+ values between 0.06 and 0.23, depending on the grid level. The flow conditions for

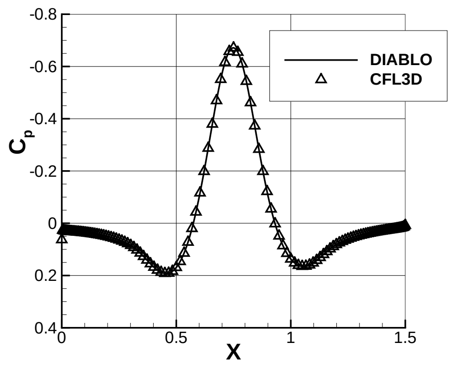


a) Lift coefficient

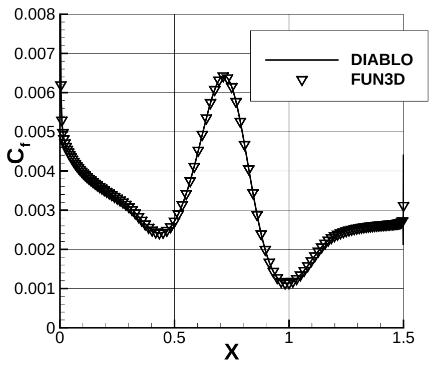


b) Drag coefficient

Fig. 15 Force coefficients for bump-in-channel flow



a) Coefficient of pressure



b) Coefficient of friction

Fig. 16 Surface coefficients for bump-in-channel flow

this case are

$$M = 0.20,\ Re = 3 \times 10^6,\ T_{\text{ref}} = 540°\text{R}.$$

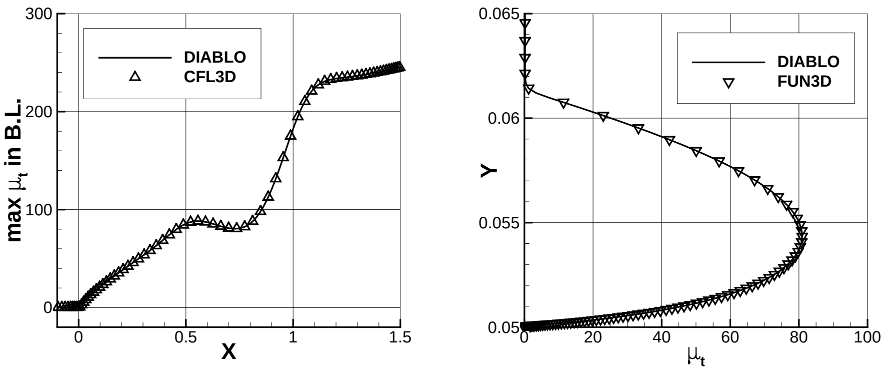
Figure 15 provides an overview of the grid convergence behaviour of lift and drag for the scalar and matrix dissipation models. For both quantities, the results obtained with DIABLO lie very close to those of the other solvers, with slightly better correspondence to the results of FUN3D. The grid convergence trends of DIABLO are in line with those of CFL3D and FUN3D.

Additionally, Figs. 16 and 17 highlight the excellent correspondence between the current algorithm and the established solvers. This is evident not only for the coefficients of pressure, C_p, and friction along the surface of the bump, but also for the value of μ_t in the boundary layer. This case verifies the implementation of DIABLO in a more complicated flow regime, with pressure gradients present in the flow due to the bump.

NACA0012 Airfoil

The final two-dimensional case considered is the flow over the NACA0012 airfoil. This case provides an opportunity not only to compare the current algorithm to CFL3D on a case of practical interest, but also to compare to the experimental data of Gregory and O'Reilly [55] (albeit at a lower Reynolds number of 3×10^6). The flow conditions are

$$M = 0.15,\ Re = 6 \times 10^6,\ T_{\text{ref}}{=}540°\text{R},\ \alpha = 0°,\ 10°,\ \text{and}\ 15°.$$



a) **Maximum μ_t in boundary layer** b) **μ_t at $x = 0.75$**

Fig. 17 μ_t distribution comparisons for bump-in-channel flow

The grid consists of 1793×513 nodes, with an off-wall spacing of 4×10^{-7} chord units. This grid represents the finest grid level available for this test case on the TMR website, and provides an average y^+ of approximately 0.1. The focus of the results for this study is the distributions of C_p and C_f on the surface of the airfoil. Experimental data are provided for C_p, and the CFL3D C_f data provided is limited to the upper surface of the airfoil.

Figure 18 presents the comparisons for all three angles of attack for the scalar dissipation model, as the two models produced nearly indistinguishable results for this grid. Due to the size of the grid, data points are omitted from the CFL3D data for increased clarity. As can be seen from the figure, the results of DIABLO provide excellent correspondence to those of CFL3D, and line up well with the experimental results. This case provides verification through comparison with CFL3D for a range of flow conditions and a validation of the solver against experimental results, including high angles of attack where boundary-layer separation is present.

Acknowledgments

The authors gratefully acknowledge financial assistance from the Natural Sciences and Engineering Research Council (NSERC), the Ontario Graduate Scholarship program, the Canada Research Chairs program, Bombardier Aerospace, and the University of Toronto.

Computations were performed on the GPC supercomputer at the SciNet HPC Consortium and the Guillimin supercomputer of the CLUMEQ consortium, both part of Compute Canada.

References

[1] Mavriplis, D. J., Vassberg, J. C., Tinoco, E. N., Mani, M., Brodersen, O. P., Eisfeld, B., Wahls, R. A., Morrison, J. H., Zickuhr, T., Levy, D., and Murayama, M., "Grid quality and resolution issues from the drag prediction workshop series," *46th AIAA Aerospace Sciences Meeting and Exhibit*, AIAA–2008–0930, Reno, Nevada, Jan. 2008.

[2] Jespersen, D., Pulliam, T., and Buning, P., "Recent enhancements to OVERFLOW (Navier-Stokes code)," *35th AIAA Aerospace Sciences Meeting and Exhibit*, AIAA–97–0644, Reno, Nevada, Jan. 1997.

[3] Nielsen, E. J., Walters, R. W., Anderson, W. K., and Keyes, D. E., "Application of Newton-Krylov methodology to a three-dimensional unstructured Euler code," *12th AIAA Computational Fluid Dynamics Conference*, AIAA–95–1733, San Diego, California, United States, June 1995.

[4] May, G. and Jameson, A., "Unstructured Algorithms for Inviscid and Viscous Flows Embedded in a Unified Solver Architecture: Flo3xx," *43rd AIAA Aerospace Sciences Meeting and Exhibit*, AIAA–2005–0318, Reno, Nevada, Jan. 2005.

[5] Mavriplis, D. J., "Grid Resolution of a Drag Prediction Workshop using the NSU3D Unstructured Mesh Solver," *17th AIAA Computational Fluid Dynamics Conference*, AIAA–2005–4729, Toronto, Canada, June 2005.

[6] Pueyo, A. and Zingg, D. W., "Efficient Newton-Krylov solver for aerodynamic computations," *AIAA Journal*, Vol. 36, No. 11, Nov. 1998, pp. 1991–1997.

[7] Chisholm, T. T. and Zingg, D. W., "A Jacobian-free Newton-Krylov algorithm for compressible turbulent fluid flows," *Journal of Computational Physics*, Vol. 228, No. 9, 2009, pp. 3490–3507.

[8] Wong, P. and Zingg, D. W., "Three-dimensional aerodynamic computations on unstructured grids using a Newton-Krylov approach," *Journal of Computers and Fluids*, Vol. 37, 2008, pp. 107–120.

[9] Hicken, J. E. and Zingg, D. W., "A parallel Newton-Krylov solver for the Euler equations discretized using simultaneous approximation terms," *AIAA Journal*, Vol. 46, No. 11, Nov. 2008, pp. 2773–2786.

[10] Osusky, M., Hicken, J. E., and Zingg, D. W., "A parallel Newton-Krylov-Schur flow solver for the Navier-Stokes equations using the SBP-SAT approach," *48th AIAA Aerospace Sciences Meeting and Aerospace Exposition*, AIAA–2010–116, Orlando, Florida, United States, Jan. 2010.

[11] Osusky, M., Boom, P. D., Del Rey Fernández, D. C., and Zingg, D. W., "An efficient Newton-Krylov-Schur parallel solution algorithm for the steady and unsteady Navier-Stokes equations," *7th International Conference on Computational Fluid Dynamics*, ICCFD7–1801, Big Island, Hawaii, USA, July 2012.

[12] Nemec, M. and Zingg, D. W., "A Newton-Krylov algorithm for Aerodynamic Design using the Navier-Stokes equations," *AIAA Journal*, Vol. 40, No. 6, 2002, pp. 1146–1154.

[13] Hicken, J. E. and Zingg, D. W., "Aerodynamic optimization algorithm with integrated geometry parameterization and mesh movement," *AIAA Journal*, Vol. 48, No. 2, Feb. 2010, pp. 401–413.

[14] Osusky, L. and Zingg, D. W., "A Novel Aerodynamic Shape Optimization Approach for Three-Dimensional Turbulent Flows," *50th AIAA Aerospace Sciences Meeting and Aerospace Exposition*, AIAA–2012–0058, Nashville, Tennessee, United States, Jan. 2012.

[15] Shu, C.-W., "High-order finite difference and finite volume WENO schemes and discontinuous Galerkin methods for CFD," *International Journal of Computational Fluid Dynamics*, Vol. 17, No. 2, 2003, pp. 107–118.

[16] Carpenter, M. H., Gottlieb, D., and Abarbanel, S., "Time-stable boundary conditions for finite-difference schemes solving hyperbolic systems: methodology and application to high-order compact schemes," *Journal of Computational Physics*, Vol. 111, No. 2, 1994, pp. 220–236.

[17] Hesthaven, J. S., "A stable penalty method for the compressible Navier-Stokes equations: III. Multi-dimensional domain decomposition schemes," *SIAM Journal on Scientific Computing*, Vol. 20, No. 1, 1998, pp. 62–93.

[18] Carpenter, M. H., Nordström, J., and Gottlieb, D., "A stable and conservative interface treatment of arbitrary spatial accuracy," *Journal of Computational Physics*, Vol. 148, No. 2, 1999, pp. 341–365.

[19] Nordström, J. and Carpenter, M. H., "High-order finite difference methods, multidimensional linear problems, and curvilinear coordinates," *Journal of Computational Physics*, Vol. 173, No. 1, 2001, pp. 149–174.

[20] Svärd, M., Carpenter, M. H., and Nordström, J., "A stable high-order finite difference scheme for the compressible Navier-Stokes equations, far-field boundary conditions," *Journal of Computational Physics*, Vol. 225, No. 1, July 2007, pp. 1020–1038.

[21] Svärd, M. and Nordström, J., "A stable high-order finite difference scheme for the compressible Navier-Stokes equations, no-slip wall boundary conditions," *Journal of Computational Physics*, Vol. 227, No. 10, May 2008, pp. 4805–4824.

[22] Nordström, J., Gong, J., van der Weide, E., and Svärd, M., "A stable and conservative high order multi-block method for the compressible Navier-Stokes equations," *Journal of Computational Physics*, Vol. 228, No. 24, 2009, pp. 9020–9035.

[23] Nordström, J. and Carpenter, M. H., "Boundary and interface conditions for high-order finite-difference methods applied to the Euler and Navier-Stokes equations," *Journal of Computational Physics*, Vol. 148, No. 2, 1999, pp. 621–645.

[24] Hicken, J. E., Osusky, M., and Zingg, D. W., "Comparison of parallel preconditioners for a Newton-Krylov flow solver," *6th International Conference on Computational Fluid Dynamics*, St. Petersburg, Russia, July 2010.

[25] Spalart, P. R. and Allmaras, S. R., "A One-Equation Turbulence Model for Aerodynamic Flows," *30th AIAA Aerospace Sciences Meeting and Exhibit*, AIAA–92–0439, Reno, Nevada, United States, Jan. 1992.

[26] Dias, S. C. and Zingg, D. W., "A high-order parallel Newton-Krylov flow solver for the Euler equations," *19th AIAA Computational Fluid Dynamics Conference*, AIAA–2009–3657, San Antonio, Texas, United States, June 2009.

[27] White, F. M., *Viscous Fluid Flow*, McGraw–Hill Book Company, New York, 1974.

[28] van Ingen, J. L., "A suggested semi-empirical method for the calculation of the boundary layer transition region," Tech. Rep. VTH–74, Delft University of Technology, 1956.

[29] Allmaras, S. R., Johnson, F. T., and Spalart, P. R., "Modifications and Clarifications for the Implementation of the Spalart-Allmaras Turbulence Model," *7th International Conference on Computational Fluid Dynamics*, ICCFD7–1902, Big Island, Hawaii, USA, July 2012.

[30] Huan, X., Hicken, J. E., and Zingg, D. W., "Interface and boundary schemes for high-order methods," *19th AIAA Computational Fluid Dynamics Conference*, AIAA–2009–3658, San Antonio, Texas, United States, June 2009.

[31] Vassberg, J. C., DeHann, M. A., Rivers, S. M., and Wahls, R. A., "Development of a Common Research Model for Applied CFD Validation Studies," *26th AIAA Applied Aerodynamics Conference*, AIAA–2008–6919, Honolulu, Hawaii, United States, Aug. 2008.

[32] Kreiss, H.-O. and Scherer, G., "Finite element and finite difference methods for hyperbolic partial differential equations," *Mathematical Aspects of Finite Elements in Partial Differential Equations*, edited by C. de Boor, Mathematics Research Center, the University of Wisconsin, Academic Press, 1974.

[33] Strand, B., "Summation by parts for finite difference approximations for d/dx," *Journal of Computational Physics*, Vol. 110, No. 1, 1994, pp. 47–67.

[34] Del Rey Fernández, D. C. and Zingg, D. W., "High-Order Compact-Stencil Summation-By-Parts Operators for the Second Derivative with Variable Coefficients," *7th International Conference on Computational Fluid Dynamics*, ICCFD7–2803, Big Island, Hawaii, USA, July 2012.

[35] Jameson, A., Schmidt, W., and Turkel, E., "Numerical solution of the Euler equations by finite volume methods using Runge-Kutta time-stepping schemes," *14th Fluid and Plasma Dynamics Conference*, AIAA–81–1259, Palo Alto, California, United States, 1981.

[36] Pulliam, T. H., "Efficient solution methods for the Navier-Stokes equations," Tech. rep., Lecture Notes for the von Kármán Inst. for Fluid Dynamics Lecture Series: Numerical Techniques for Viscous Flow Computation in Turbomachinery Bladings, Rhode-Saint-Genèse, Belgium, Jan. 1986.

[37] Swanson, R. C. and Turkel, E., "On central-difference and upwind schemes," *Journal of Computational Physics*, Vol. 101, No. 2, 1992, pp. 292–306.

[38] Mattsson, K., Svärd, M., and Shoeybi, M., "Stable and accurate schemes for the compressible Navier-Stokes equations," *Journal of Computational Physics*, Vol. 227, No. 4, 2008, pp. 2293–2316.

[39] Diener, P., Dorband, E. N., Schnetter, E., and Tiglio, M., "Optimized high-order derivative and dissipation operators satisfying summation by parts, and application in three-dimensional multi-block evolutions," *Journal of Scientific Computing*, Vol. 32, No. 1, 2007, pp. 109–145.

[40] Mattsson, K. and Nordström, J., "Summation by parts operators for finite-difference approximations of second derivatives," *Journal of Computational Physics*, Vol. 199, No. 2, 2004, pp. 503–540.

[41] Spalart, P. R. and Rumsey, C. L., "Effective inflow conditions for turbulence models in aerodynamic calculations," *AIAA Journal*, Vol. 45, No. 10, 2007, pp. 2544–2553.

[42] Lomax, H., Pulliam, T. H., and Zingg, D. W., *Fundamentals of Computational Fluid Dynamics*, Springer–Verlag, Berlin, Germany, 2001.

[43] Nichols, J. and Zingg, D. W., "A three-dimensional multi-block Newton-Krylov flow solver for the Euler equations," *17th AIAA Computational Fluid Dynamics Conference*, AIAA–2005–5230, Toronto, Canada, June 2005.

[44] Blanco, M. and Zingg, D. W., "Fast Newton-Krylov method for unstructured grids," *AIAA Journal*, Vol. 36, No. 4, April 1998, pp. 607–612.

[45] Kim, D. B. and Orkwis, P. D., "Jacobian update strategies for quadratic and near-quadratic convergence of Newton and Newton-like implicit schemes," *31st AIAA Aerospace Sciences Meeting and Exhibit*, AIAA–93–0878, Reno, Nevada, 1993.

[46] Keyes, D. E., "Aerodynamic applications of Newton-Krylov-Schwarz solvers," *Proceedings of the 14th International Conference on Numerical Methods in Fluid Dynamics*, Springer, New York, 1995, pp. 1–20.

[47] Saad, Y. and Sosonkina, M., "Distributed Schur complement techniques for general sparse linear systems," *SIAM Journal of Scientific Computing*, Vol. 21, No. 4, 1999, pp. 1337–1357.

[48] Mulder, W. A. and van Leer, B., "Experiments with implicit upwind methods for the Euler equations," *Journal of Computational Physics*, Vol. 59, No. 2, 1985, pp. 232–246.

[49] Meijerink, J. A. and van der Vorst, H. A., "An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix," *Mathematics of Computation*, Vol. 31, No. 137, Jan. 1977, pp. 148–162.

[50] Nordström, J., Eriksson, S., and Eliasson, P., "Weak and strong wall boundary procedures and convergence to steady-state of the Navier-Stokes equations," *Journal of Computational Physics*, Vol. 231, No. 14, 2012, pp. 4867–4884.

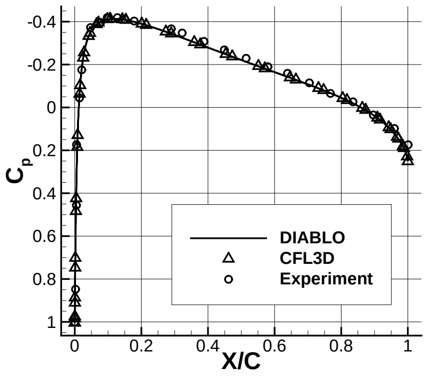
[51] Schmitt, V. and Charpin, F., "Pressure distributions on the ONERA-M6-wing at transonic Mach numbers," Tech. rep., Office National d'Etudes et Recherches Aerospatiales, 92320, Chatillon, France, 1979.

[52] Vassberg, J. C., "A Unified Baseline Grid about the Common Research Model Wing-Body for the Fifth AIAA CFD Drag Prediction Workshop," *29th AIAA Applied Aerodynamics Conference*, AIAA–2011–3508, Honolulu, Hawaii, United States, June 2011.

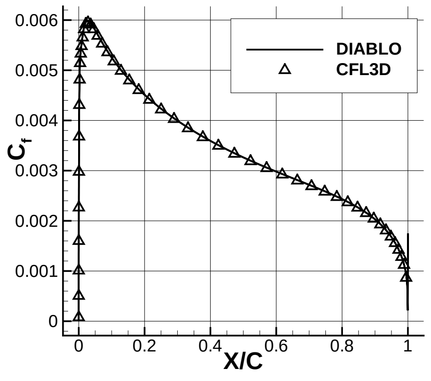
[53] Apponsah, K. P. and Zingg, D. W., "A Load Balancing Tool for Structured Multi-Block Grid CFD Applications," *20th Annual Conference of the CFD Society of Canada*, Canmore, Alberta, Canada, May 2012.

[54] Baker, T. J., "Mesh generation: Art or science?" *Progress in Aerospace Sciences*, Vol. 41, No. 1, 2005, pp. 29–63.

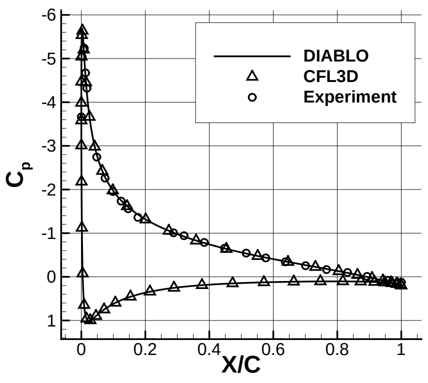
[55] Gregory, N. and O'Reilly, C. L., "Low-Speed Aerodynamic Characteristics of NACA 0012 Aerofoil Sections, including the Effects of Upper-Surface Roughness Simulation Hoar Frost," Tech. Rep. NASA R&M 3276, 1970.



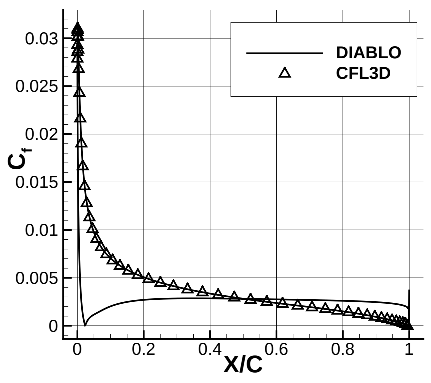
a) C_p at $\alpha = 0°$



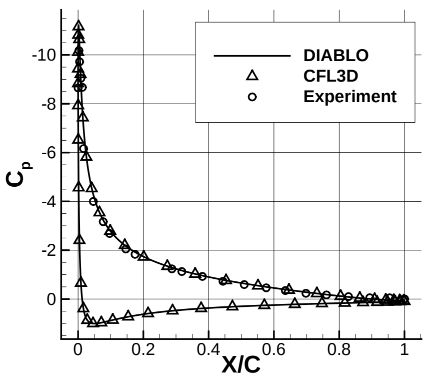
b) C_f at $\alpha = 0°$



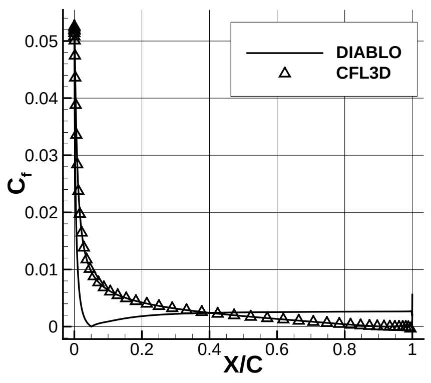
c) C_p at $\alpha = 10°$



d) C_f at $\alpha = 10°$



e) C_p at $\alpha = 15°$



f) C_f at $\alpha = 15°$

Fig. 18 C_p and C_f comparison for NACA0012 flows