A parallel Newton-Krylov-Schur flow solver for the Reynolds-Averaged Navier-Stokes equations

Michal Osusky* and David W. Zingg†

Institute for Aerospace Studies, University of Toronto, Toronto, Ontario, M3H 5T6, Canada

This paper presents a parallel Newton-Krylov flow solution algorithm for the three-dimensional Navier-Stokes equations coupled with the Spalart-Allmaras one-equation turbulence model. The algorithm employs summation-by-parts operators on multi-block structured grids, while simultaneous approximation terms are used to enforce boundary conditions and coupling at block interfaces. The discrete equations are solved iteratively with an inexact-Newton method, and the linear system of each Newton iteration is solved using the flexible GMRES Krylov subspace iterative method with the approximate-Schur parallel preconditioner. The algorithm performs well at a multitude of flow conditions, ranging from subsonic to transonic flow. A grid convergence study shows the efficiency of the flow solver at various grid refinements, as well as excellent correspondence of the grid converged aerodynamic force coefficients in comparison to an established flow solver. A transonic solution around the ONERA M6 wing on a mesh with 15 million nodes shows good agreement with experiment. The residual is reduced by twelve orders of magnitude in 89 minutes on 128 processors. The solution of transonic flow over the Common Research Model wing-body-horizontal-tail geometry shows good agreement with other computed solutions. Parallel scaling results using up to 4096 processors demonstrate the excellent scaling capability of the algorithm.

I. Introduction

Rising fuel prices and increasing concern over the impact of aviation on the environment has led the aviation industry to seek ever more efficient designs for aircraft. One of the avenues by which efficiency can be improved is by decreasing drag. To this end, aerodynamic shape optimization algorithms can be used to optimize existing aerodynamic shapes, or even come up with novel shapes. At the core of such an algorithm lies a flow solver, but with the increasing complexity and size of three-dimensional optimization problems, and the computational cost this carries with it, it is critical that the flow solver be as fast and efficient as possible. This paper presents the description of a flow solution algorithm for three-dimensional turbulent flows that is well suited for use within an optimization algorithm.

In the computation of high-Reynolds number turbulent flows over complex geometries, current approaches span the use of structured and unstructured grids, finite-volume or finite-difference approximations, and a plethora of combinations of linear solvers and preconditioners. A few examples of popular RANS solvers are OVERFLOW,[1] FUN3D,[2] Flo3xx,[3] and NSU3D.[4] This paper presents an efficient parallel three-dimensional multi-block structured solver for turbulent flows over aerodynamic geometries, extending previous work[5,6,7] on an efficient parallel Newton-Krylov flow solver for the Euler equations and the Navier-Stokes equations in the laminar flow regime.

In order to accommodate complex three-dimensional shapes, the multi-block approach, which breaks the computational domain into several subdomains, is used. This approach has the added benefit of being easily utilized in a parallel solution algorithm, since each block can be assigned to an individual process. 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

*PhD Candidate, AIAA Student Member

†Professor and Director, Tier 1 Canada Research Chair in Computational Fluid Dynamics, J. Armand Bombardier Foundation Chair in Aerospace Flight, Associate Fellow AIAA

in an accurate and time-stable manner,[8] and later extended to deal with block interfaces.[9,10,11] Svärd *et al.*[12,13] and Nordström *et al.*[14] 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, reduces the communication for parallel algorithms, and ensures linear time stability when coupled with summation-by-parts (SBP) operators. However, SATs have received limited use in computational aerodynamics applications, and there are only a few demonstrations of their use for practical aerodynamic problems.[5,6,7,15] They present a difficulty in that they can necessitate the use of small time steps with explicit solvers.[16] Hence, the combination of SATs with a parallel Newton-Krylov solver has the potential to be an efficient approach. Parallel preconditioning is a critical component of a scalable Newton-Krylov algorithm. Hicken *et al.*[17] have shown that the approximate-Schur preconditioner scales well to at least 1000 processors when inviscid and laminar flows are considered. The objective of this paper is to extend the combination 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,[18] to produce an efficient parallel solver for turbulent flows.

The paper is divided into the following sections. Section II presents an overview of the governing equations, while Section III presents the spatial discretization used, including the SATs at block interfaces and boundaries, as it applies to the turbulence model. Section IV provides details of the Newton-Krylov-Schur method used to solve the large nonlinear system resulting from the discretization of the Navier-Stokes equations, including special considerations for the turbulence model. In Section V, grid convergence studies and convergence histories are presented, as well as results comparing the flow solutions produced by the new algorithm against results obtained from an established flow solver and experimental data. The parallel scaling performance of the algorithm is also highlighted in this section. Conclusions are given in Section VI.

II. Governing Equations

A. The Navier-Stokes Equations

The three-dimensional Navier-Stokes equations are given by

$$\partial_t \mathbf{Q} + \partial_x \mathbf{E} + \partial_y \mathbf{F} + \partial_z \mathbf{G} = \frac{1}{\mathrm{Re}}\Big(\partial_x \mathbf{E_v} + \partial_y \mathbf{F_v} + \partial_z \mathbf{G_v}\Big), \tag{1}$$

where $\mathrm{Re} = \frac{\rho_\infty a_\infty l}{\mu_\infty}$,

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

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

Additionally, ρ is the density, a the sound speed, e the energy, p the pressure, l the chord length, μ the viscosity, u, v, and w the Cartesian velocity components, and τ the Newtonian stress tensor. The '∞' subscript denotes a free-stream value for the given quantity. 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:[19]

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

where $S^* = 198.6^\circ\mathrm{R}$ and $T_\infty = 460^\circ\mathrm{R}$.

Applying the coordinate transformation $(x, y, z) \rightarrow (\xi, \eta, \zeta)$, which allows us to treat the governing equations on a uniform computational grid, the Navier-Stokes equations can be rewritten as

$$\partial_t \hat{\mathbf{Q}} + \partial_\xi \hat{\mathbf{E}} + \partial_\eta \hat{\mathbf{F}} + \partial_\zeta \hat{\mathbf{G}} = \frac{1}{\mathrm{Re}}\Big(\partial_\xi \hat{\mathbf{E}}_\mathbf{v} + \partial_\eta \hat{\mathbf{F}}_\mathbf{v} + \partial_\zeta \hat{\mathbf{G}}_\mathbf{v}\Big), \tag{3}$$

where

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

and J is the metric Jacobian that results from the coordinate transformation. The notation ξ_x, for example, is a shorthand form of $\partial_x \xi$. The viscous stresses also undergo the coordinate transformation and result in the following expressions:

$$\begin{aligned}
\tau_{xx} &= \tfrac{4}{3}(\mu+\mu_t)(\xi_x u_\xi + \eta_x u_\eta + \zeta_x u_\zeta) - \tfrac{2}{3}(\mu+\mu_t)\left(\xi_y v_\xi + \eta_y v_\eta + \zeta_y v_\zeta + \xi_z w_\xi + \eta_z w_\eta + \zeta_z w_\zeta\right) \\
\tau_{xy} &= (\mu+\mu_t)\left(\xi_y u_\xi + \eta_y u_\eta + \zeta_y u_\zeta + \xi_x v_\xi + \eta_x v_\eta + \zeta_x v_\zeta\right) \\
\tau_{xz} &= (\mu+\mu_t)\left(\xi_z u_\xi + \eta_z u_\eta + \zeta_z u_\zeta + \xi_x w_\xi + \eta_x w_\eta + \zeta_x w_\zeta\right) \\
\tau_{yx} &= \tau_{xy} \\
\tau_{yy} &= \tfrac{4}{3}(\mu+\mu_t)(\xi_y v_\xi + \eta_y v_\eta + \zeta_y v_\zeta) - \tfrac{2}{3}(\mu+\mu_t)\left(\xi_x u_\xi + \eta_x u_\eta + \zeta_x u_\zeta + \xi_z w_\xi + \eta_z w_\eta + \zeta_z w_\zeta\right) \\
\tau_{yz} &= (\mu+\mu_t)\left(\xi_z v_\xi + \eta_z v_\eta + \zeta_z v_\zeta + \xi_y w_\xi + \eta_y w_\eta + \zeta_y w_\zeta\right) \\
\tau_{zx} &= \tau_{xz} \\
\tau_{zy} &= \tau_{yz} \\
\tau_{zz} &= \tfrac{4}{3}(\mu+\mu_t)(\xi_z w_\xi + \eta_z w_\eta + \zeta_z w_\zeta) - \tfrac{2}{3}(\mu+\mu_t)\left(\xi_x u_\xi + \eta_x u_\eta + \zeta_x u_\zeta + \xi_y v_\xi + \eta_y v_\eta + \zeta_y v_\zeta\right) \\
E_{v,5} &= u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + (\mu \mathrm{Pr}^{-1} + \mu_t \mathrm{Pr}_t^{-1})(\gamma-1)^{-1}[\xi_x\partial_\xi(a^2) + \eta_x\partial_\eta(a^2) + \zeta_x\partial_\zeta(a^2)] \\
F_{v,5} &= u\tau_{yx} + v\tau_{yy} + w\tau_{yz} + (\mu \mathrm{Pr}^{-1} + \mu_t \mathrm{Pr}_t^{-1})(\gamma-1)^{-1}[\xi_y\partial_\xi(a^2) + \eta_y\partial_\eta(a^2) + \zeta_y\partial_\zeta(a^2)] \\
G_{v,5} &= u\tau_{zx} + v\tau_{zy} + w\tau_{zz} + (\mu \mathrm{Pr}^{-1} + \mu_t \mathrm{Pr}_t^{-1})(\gamma-1)^{-1}[\xi_z\partial_\xi(a^2) + \eta_z\partial_\eta(a^2) + \zeta_z\partial_\zeta(a^2)]
\end{aligned}$$

where μ_t is the eddy viscosity (also referred to as turbulent viscosity), Pr and Pr_t are the laminar and turbulent Prandtl numbers, taken as 0.72 and 0.90, respectively, and $\gamma = 1.4$ for air.

B. Spalart-Allmaras One-Equation Turbulence Model

The viscous terms in the Navier-Stokes equations contain a turbulent viscosity component which has not yet been defined. In order to obtain this value, the Spalart-Allmaras one-equation turbulence model[18] is used. The model solves a transport equation for a turbulence-like variable, $\tilde{\nu}$, that is related to the turbulent viscosity term required for the Navier-Stokes equations. The model itself is a sixth equation that is solved concurrently with the five Navier-Stokes equations.

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

$$\begin{aligned}
\frac{\partial \tilde{\nu}}{\partial t} + u_i \frac{\partial \tilde{\nu}}{\partial x_i} =& \frac{c_{b1}}{\mathrm{Re}}[1 - f_{t2}]\tilde{S}\tilde{\nu} + \frac{1 + c_{b2}}{\sigma_\mathrm{t}\mathrm{Re}}\nabla\cdot[(\nu+\tilde{\nu})\nabla\tilde{\nu}] - \frac{c_{b2}}{\sigma_\mathrm{t}\mathrm{Re}}(\nu+\tilde{\nu})\nabla^2\tilde{\nu} \\
&- \frac{1}{\mathrm{Re}}\left[c_{w1}f_w - \frac{c_{b1}}{\kappa^2}f_{t2}\right]\left(\frac{\tilde{\nu}}{d}\right)^2.
\end{aligned} \tag{4}$$

The spatial derivatives on the left side of the equation represent advection. The first term on the right side represents production, and the fourth term represents destruction. The second and third terms account for dissipation. Since the flow used in all simulations is considered fully turbulent, the f_{t1} trip terms of the original model are omitted.

In order to fully define the model, several relations and constants need to be specified. In the current

implementation, the following are used:

$$\begin{aligned}
\mu_t &= \rho\nu_t, \\
\nu_t &= f_{v1}\tilde{\nu}, \\
f_{v1} &= \frac{\chi^3}{\chi^3 + c_{v1}^3}, \\
\chi &= \tilde{\nu}/\nu, \\
S &= \left[\left(\frac{\partial w}{\partial y} - \frac{\partial v}{\partial z}\right)^2 + \left(\frac{\partial u}{\partial z} - \frac{\partial w}{\partial x}\right)^2 + \left(\frac{\partial v,}{\partial x} - \frac{\partial u}{\partial y}\right)^2\right]^{-\frac{1}{2}}, \\
\tilde{S} &= \mathrm{Re}S + \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2}, \\
f_{v2} &= 1 - \frac{\chi}{1 + \chi f_{v1}}, \\
f_w &= g\left[\frac{1 + c_{w3}^6}{g^6 + c_{w3}^6}\right]^{\frac{1}{6}}, \\
g &= r + c_{w2}\left(r^6 - r\right), \\
r &= \min\left(\frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2}, 10\right), \\
f_{t2} &= c_{t3}\exp\left(-c_{t4}\chi^2\right),
\end{aligned}$$

with $c_{b1} = 0.1355$, $c_{b2} = 0.622$, $\sigma_t = 2/3$, $c_{v1} = 7.1$, $c_{w2} = 0.3$, $\kappa = 0.41$, $c_{w3} = 2.0$, $c_{w1} = \frac{c_{b1}}{\kappa^2} + \frac{1}{\sigma_t}(1 + c_{b2})$, $c_{v2} = 5.0$, $c_{t3} = 1.2$, and $c_{t4} = 0.5$. The off-wall distance at each computational node is denoted by d. Additionally, precautions are taken to ensure that neither the vorticity, S, nor $\tilde{S}$ approaches zero or becomes negative, which could lead to numerical problems.

The complete details of the coordinate transformation for the turbulence model can be found in the Appendix.

III. Spatial Discretization

The spatial discretization of the Navier-Stokes equations and the turbulence model is obtained by the use of second-order SBP operators, while inter-block coupling and boundary conditions are enforced by the use of SATs. As details of the discretization of the Navier-Stokes equations were presented previously,[7,20] this section will focus exclusively on the application of the SBP-SAT approach to the Spalart-Allmaras one-equation turbulence model. For more details, including the theory behind the development of the SBP-SAT approach, please refer to references 5, 7, 12, 13, 14, 21, and 22.

The numerical dissipation used in the present work employs the scalar dissipation model developed by Jameson *et al.*[23] and later refined by Pulliam.[24] The model consists of second- and fourth-difference dissipation operators, whose magnitudes are controlled by the κ_2 and κ_4 coefficients, respectively.

A. Summation-by-Parts Operators

SBP operators allow for the construction of finite difference approximations to the spatial derivatives that appear in the governing equations. In particular, the turbulence model includes both first and second derivatives in all three coordinate directions, all of which require the application of the appropriate difference operator. In order to highlight the application of the SBP operators to the turbulence model, the various spatial derivatives will be treated individually.

1. Advective terms

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

$$U\frac{\partial \tilde{\nu}}{\partial \xi}, \tag{5}$$

where U is the contravariant velocity given by $\xi_x u + \xi_y v + \xi_z w$.

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 equated to a centered difference operator added to a dissipation operator. In detail, the derivative can be taken as

$$U\frac{\partial\tilde{\nu}}{\partial\xi} \approx \mathbf{U}\mathcal{H}^{-1}\mathcal{Q}\tilde{\boldsymbol{\nu}} + \frac{1}{2}|\mathbf{U}|\mathcal{H}^{-1}\mathcal{D}_d^T\mathcal{D}_d\tilde{\boldsymbol{\nu}} \tag{6}$$

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

$$\mathbf{U} = \mathrm{diag}\left(U_1, U_2, ..., U_N\right), \quad |\mathbf{U}| = \mathrm{diag}\left(|U_1|, |U_2|, ..., |U_N|\right)$$

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

$$\mathcal{D}_d = \begin{bmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \\ & & & & 0 \end{bmatrix},$$

and N is the number of nodes in ξ-direction. The spatial parameter h takes on the value of the spatial difference 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 unity for all three coordinate directions. 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:	$\frac{1}{2}U_i\left(\tilde{\nu}_{i+1} - \tilde{\nu}_{i-1}\right) - \frac{1}{2}\lvert U_i\rvert\left(\tilde{\nu}_{i+1} - 2\tilde{\nu}_i + \tilde{\nu}_{i-1}\right)$
high side:	$\left(U_N + \lvert U_N\rvert\right)\left(\tilde{\nu}_N - \tilde{\nu}_{N-1}\right).$

The first derivative operator present in (6), written as $\mathcal{D}_1 = \mathcal{H}^{-1}\mathcal{Q}$, is also used for the derivatives present in the vorticity term, S.

2. Diffusive terms

As with the viscous terms present in the Navier-Stokes equations, the diffusive terms appearing in the turbulence model are composed of double derivatives, possessing the general form

$$\partial_\xi\left(b\partial_\xi\left(\tilde{\nu}\right)\right), \tag{7}$$

where b is a spatially varying coefficient.

An SBP operator for such a term, denoted $\mathcal{D}_2$, was presented by Mattsson *et al.*:[25]

$$\mathcal{D}_2 = \mathcal{H}^{-1}\left(-\mathcal{D}^T\tilde{\mathcal{B}}\mathcal{D} + \mathcal{B}\mathcal{S}\right), \tag{8}$$

where

$$\mathcal{D} = \frac{1}{h}\begin{bmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & & \\ & & & -1 & 1 \\ & & & -1 & 1 \end{bmatrix}, \quad \tilde{\mathcal{B}} = \frac{h}{2}\begin{bmatrix} b_0 + b_1 & & & & \\ & b_1 + b_2 & & & \\ & & \ddots & & \\ & & & b_{N-1} + b_N & \\ & & & & 0 \end{bmatrix},$$

$$\mathcal{B} = \begin{bmatrix} -b_0 & & & & \\ & 0 & & & \\ & & \ddots & & \\ & & & 0 & \\ & & & & b_N \end{bmatrix}, \quad \mathcal{S} = \frac{1}{h} \begin{bmatrix} -\frac{3}{2} & 2 & -\frac{1}{2} & & & \\ & 1 & & & & \\ & & \ddots & & & \\ & & & & 1 & \\ & & & \frac{1}{2} & -2 & \frac{3}{2} \end{bmatrix}.$$

For an internal node, this will result in the narrow stencil used by Pulliam[24] (with k and m subscripts suppressed),

$$\partial_\xi(b\partial_\xi\tilde{\nu})_j \approx \frac{1}{2}(b_{j+1}+b_j)(\tilde{\nu}_{j+1}-\tilde{\nu}_j) - \frac{1}{2}(b_j+b_{j-1})(\tilde{\nu}_j-\tilde{\nu}_{j-1}). \tag{9}$$

At the high- and low-side boundaries, where one-sided differences are employed, the discretization takes on the form

$$\begin{aligned} \partial_\xi(b\partial_\xi\tilde{\nu})_{j=1} &\approx -b_j\left[2(\tilde{\nu}_{j+1}-\tilde{\nu}_j)-(\tilde{\nu}_{j+2}-\tilde{\nu}_{j+1})\right] + b_{j+1}(\tilde{\nu}_{j+1}-\tilde{\nu}_j), \\ \partial_\xi(b\partial_\xi\tilde{\nu})_{j=N} &\approx b_j\left[2(\tilde{\nu}_j-\tilde{\nu}_{j-1})-(\tilde{\nu}_{j-1}-\tilde{\nu}_{j-2})\right] - b_{j-1}(\tilde{\nu}_j-\tilde{\nu}_{j-1}). \end{aligned} \tag{10}$$

B. Simultaneous Approximation Terms

The use of SBP operators ties in closely to the application of SAT penalties at block boundaries, be they interfaces or domain boundaries. SATs are used to preserve inter-block continuity, or enforce specific boundary conditions. The development of the SATs was carried out in a manner that parallels the use of SATs for the Navier-Stokes equations, leading to both the advective (Euler-like) SATs, as well as diffusive (viscous-like) SATs. As with the mean-flow SATs, the terms presented here are added to the right-hand-side of equation (4). For simplicity, only SATs for interfaces normal to the ξ-direction will be shown, but all other required SATs can be obtained by simply making appropriate changes to the contravariant velocities and grid metrics that appear in the SAT expressions.

1. Advective terms

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 presented in reference 5. This can be achieved using the following form of the SAT:

$$\mathrm{SAT}_{\mathrm{adv}} = \mathcal{H}_b^{-1}\sigma_{\mathrm{a}}\left(\tilde{\nu}_{\mathrm{local}} - \tilde{\nu}_{\mathrm{target}}\right), \tag{11}$$

where $\mathcal{H}_b$ is the boundary node element of the diagonal norm matrix $\mathcal{H}$, $\tilde{\nu}_{\mathrm{local}}$ 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,

$$\sigma_{\mathrm{a}} = -\frac{1}{2}\left[\max\left(|U|,\phi\right) + \delta_{\mathrm{a}}U\right], \tag{12}$$

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

$$\phi = V_l\left(|U| + a\sqrt{\xi_x^2+\xi_y^2+\xi_z^2}\right), \tag{13}$$

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.

2. Diffusive terms

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

$$\mathrm{SAT}_{\mathrm{diff\_flux}} = \mathcal{H}_b^{-1}\sigma_{\mathrm{df}}\left(g_{\mathrm{local}} - g_{\mathrm{target}}\right), \tag{14}$$

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

$$g = \frac{1}{\sigma_{\mathrm{t}}\mathrm{Re}}\left(\nu + \tilde{\nu}\right)\left(\xi_x^2 + \xi_y^2 + \xi_z^2\right)\delta_\xi\tilde{\nu}, \tag{15}$$

where $\delta_\xi\tilde{\nu}$ is a one-sided first derivative consistent with the definition of the double derivative SBP operator at block boundaries, specified in the $\mathcal{S}$ matrix of equation (8). The parameter σ_{t} is defined as part of the turbulence model with a value of 2/3. The gradient difference SAT is applied at the farfield boundary, where the target gradient is set to 0, or at block interfaces, where the target gradient is 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.*[14] for the Navier-Stokes equations,

$$\mathrm{SAT}_{\mathrm{diff\_vars}} = -\mathcal{H}_b^{-1}\frac{1}{4\sigma_{\mathrm{t}}\mathrm{Re}}\sigma_{\mathrm{dv}}\left(\tilde{\nu}_{\mathrm{local}} - \tilde{\nu}_{\mathrm{target}}\right), \tag{16}$$

where

$$\sigma_{\mathrm{dv}} = \left(\nu + \tilde{\nu}\right)\left(\xi_x^2 + \xi_y^2 + \xi_z^2\right). \tag{17}$$

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).

3. Production and destruction terms

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. The production and destruction terms have no physical meaning for these nodes, and numerically they are undefined due to a division by a zero off-wall distance. However, a lack of any source term for the surface nodes leads to a significant difference in the residual between the surface nodes and the nodes directly above the surface. This difference often results in large, destabilizing updates to the turbulence variable, often causing the code to diverge.

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 $d = d_{\mathrm{min}}/2$, 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 $\tilde{\nu}$ towards the target value of 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:

$$\mathcal{R}(\mathcal{Q}) = 0, \tag{18}$$

where $\mathcal{Q}$ represents the complete solution vector. When time-marched with the implicit Euler time-marching method and a local time linearization, this system results in a large set of linear equations of the form:[26]

$$\left(\frac{\mathcal{I}}{\Delta t} + \mathcal{A}^{(n)}\right)\Delta\mathcal{Q}^{(n)} = -\mathcal{R}^{(n)}, \tag{19}$$

where n is the outer (nonlinear) iteration index, Δt is the time step, $\mathcal{I}$ is the identity matrix, $\mathcal{R}^{(n)} = \mathcal{R}(\mathcal{Q}^{(n)})$, $\Delta\mathcal{Q}^{(n)} = \mathcal{Q}^{(n+1)} - \mathcal{Q}^{(n)}$, and

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

is the Jacobian.

In the infinite time step limit, the above describes Newton's method and will converge quadratically if a suitable initial iterate, $\mathcal{Q}^{(0)}$, is known. This initial iterate must be sufficiently close to the solution of (18). 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. It should be noted that both phases result in a large set of linear equations at each outer iteration, which are solved to a specified tolerance using the preconditioned Krylov iterative solver GMRES.

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 Jacobian matrix, $\mathcal{A}_1$, has been shown to be an effective replacement of the true Jacobian, $\mathcal{A}$, during the start-up phase.[27,28,29] 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

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

where σ is a lumping factor. A value of $\sigma = 8$ has been shown to work well for the Navier-Stokes solutions with scalar dissipation.[30] The modified fourth-difference dissipation coefficient, $\tilde{\kappa}_4$, is set to zero. Applying this lumping approach reduces the accuracy of the Jacobian to first-order. This does not impact the accuracy of the final solution, but, more importantly, it 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 term 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 (see Appendix).

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:

$$\Delta t^{(n)}_{j,k,m} = \frac{J_{j,k,m}\Delta t^{(n)}_{\text{ref}}}{1 + \sqrt[3]{J_{j,k,m}}}, \tag{20}$$

where (j,k,m) denote the computational coordinates 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 (20). The reference time step is

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

where typical values used for turbulent flow solutions are $a = 0.001$ and $b = 1.3$.

Once formed, the first-order Jacobian is factored using block incomplete lower-upper factorization (BILU) with fill level p in order to construct the preconditioner used throughout the solution process. This is a

computationally expensive task, especially in the approximate-Newton phase, which requires many outer iterations. Previous work[5,31] has shown that lagging the update of the preconditioner (freezing it for a number of iterations) during the start-up phase can positively impact the efficiency of the flow solver. A typical fill level for the factorization is 2.

Effective preconditioning is critical in creating an efficient parallel linear solver. Two approaches to parallel preconditioning, namely additive-Schwarz[32] and approximate-Schur,[33] have been previously investigated in the context of a parallel Newton-Krylov flow solver for the Euler[5,17] and Navier-Stokes[17] 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.

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:

$$R_d^{(n)} \equiv \frac{||\mathcal{R}^{(n)}||_2}{||\mathcal{R}^{(0)}||_2}. \tag{21}$$

For turbulent flows, once this value reaches 0.001, i.e. the residual has dropped by 3 orders of magnitude in the approximate-Newton phase, the algorithm switches to the inexact-Newton method. This initial drop is larger than the one 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, $\mathcal{R}^{(0)}$, begins with 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. For example, transonic flows can require longer start-up phases.

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 diagonal of 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,[34] by which a new reference time step is calculated and used in (20):

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

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 method, this method 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

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

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:

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

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.

C. Special Considerations for Turbulence Model

The addition of the turbulence model to the linear system of (19) 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 the work done by Chisholm and Zingg,[35] 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 10^{-3}. 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. In order to normalize the flow variable values, the turbulence variable quantity is also multiplied by 10^{-3}. Hence, instead of solving the system presented in (19), the solution algorithm tackles a scaled system of the form

$$S_a S_r \left(\frac{\mathcal{I}}{\Delta t} + \mathcal{A}^{(n)} \right) S_c S_c^{-1} \Delta \mathcal{Q}^{(n)} = -S_a S_r \mathcal{R}^{(n)}, \tag{22}$$

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 magnitudes 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

$$S_r = \mathrm{diag}\left(S_{r1}, S_{r2}, ..., S_{rN}\right), \quad S_c = \mathrm{diag}\left(S_{c1}, S_{c2}, ..., S_{cN}\right),$$

where

$$S_{ri} = \begin{bmatrix} J_i^{2/3} & & & & & & \\ & J_i^{2/3} & & & & & \\ & & J_i^{2/3} & & & & \\ & & & J_i^{2/3} & & & \\ & & & & J_i^{2/3} & & \\ & & & & & 10^{-3} J_i^{-1/3} \end{bmatrix}, \quad S_{ci} = \begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & 1 & & & & \\ & & & 1 & & & \\ & & & & 1 & & \\ & & & & & 10^3 \end{bmatrix},$$

and J_i is the value of the metric Jacobian at the i^{th} node in the computational domain. The values in the auto-scaling matrix are calculated based on the equation-wise residual L_2-norms of the partially scaled system $S_r \mathcal{R}^{(n)}$, and are identical for each node in the domain. Instead, they scale the individual component equations by different amounts. Any residual values required for the time step calculation make use of the partially scaled residual $S_r \mathcal{R}^{(n)}$.

During the 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 $10^{-14}\mu/\rho$, while all other locations are trimmed to $10^{-3}\mu/\rho$. The local μ/ρ term is introduced such that advective and diffusive fluxes do not vanish completely from regions where several adjacent nodes are trimmed during the same iteration.

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 $8.5e-10$. Finally, the vorticity-like term, $\tilde{S}$, cannot be

Table 1: 2D grid parameters for grid convergence study

Grid	Grid Size	Nodes on airfoil	Nodes in wake	Off-wall spacing
coarse	166×39	116	26	1.15e-6
medium	331×77	231	51	5.13e-7
fine	661×153	461	101	2.43e-7

allowed to reach zero or become negative, which would have a destabilizing effect on the values of the production and destruction terms. We have found that preventing this value from becoming smaller than 10^{-5}Ma works well, where Ma is the farfield Mach number.

The farfield condition used with the turbulence model sets the target farfield value of $\tilde{\nu}$ to 3.0, as suggested by Spalart and Rumsey.[36] The target surface value of $\tilde{\nu}$ is set to 0.0. Furthermore, the turbulence quantity is initialized to the farfield value at the beginning of the flow solution process.

V. Results

The results presented in this section highlight the use of the SBP-SAT approach in solving the Reynolds-averaged Navier-Stokes equations in either two or three dimensions, as well as the performance of the parallel Newton-Krylov-Schur algorithm. Two-dimensional solutions are presented in order to highlight the accuracy and robustness of the current solver, DIABLO, as compared to flow solutions obtained using an extensively verified and validated two-dimensional finite-difference Newton-Krylov algorithm, OPTIMA2D.[37] The spatial discretization in OPTIMA2D is essentially the same as that in the well-known ARC2D.[24] Additionally, a grid convergence study for the NACA 0012 airfoil is performed, again comparing to the solutions provided by OPTIMA2D. Convergence histories are presented to highlight the efficiency of the Newton-Krylov-Schur algorithm in obtaining fully converged steady-state flow solutions.

A comparison to three-dimensional experimental solutions is made for a transonic flow over the ONERA M6 wing and the Common Research Model[38] (CRM) wing-body-horizontal-tail configuration. Finally, the parallel scaling of the algorithm is investigated by computing a subsonic flow solution around the same geometry with up to 4096 processors.

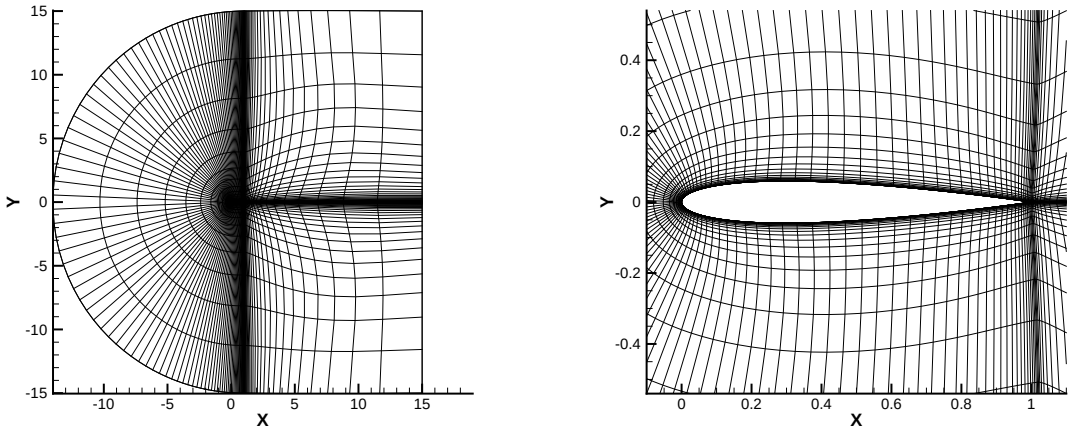
A. Two-Dimensional Solutions

A grid convergence study was performed on a series of successively coarsened grids. The finest grid was created using an elliptical grid generation program around the NACA 0012 airfoil geometry. The two coarser grids were created by removing every second grid node in both coordinate directions. The grid topology is a single-block C-mesh, with a single interface present at the wake-cut. Table 1 provides a summary of the grid characteristics for the three meshes, while Figure 1 shows the coarsest grid with a detail of the mesh around the airfoil. The flow conditions used for this test were

$$\text{Ma} = 0.30, \text{Re} = 3.00\text{e}6, \alpha = 2.0^\circ,$$

where α is the angle of attack.

Converged steady-state flow solutions were obtained for all grids by reducing the initial flow residual by 12 orders of magnitude. Table 2 compares the lift and drag coefficient values, C_l and C_d, produced by OPTIMA2D and DIABLO, along with grid converged values of the two coefficients, F^*, and their order of convergence, p, calculated using Richardson extrapolation.[39] The grid converged values of C_l and C_d show excellent correspondence between the two solvers, with any discrepancy due to the minor differences in the spatial discretizations used in the two solvers and the use of the thin-layer approximation in OPTIMA2D, as well as the treatment of boundary conditions. Both solvers exhibit the expected second order convergence for force coefficients. The values C_l and C_d on each grid level, as compared to the grid converged values, exhibits similar behaviour for both solvers, with the finest grid result for C_d being within 1.4% for OPTIMA2D and 0.7% for DIABLO.



(a) complete grid

(b) detail of grid near airfoil

Figure 1: Coarse grid used in grid convergence study

Table 2: Grid convergence values for C_l and C_d

Solver		C_l	C_l % difference to F^*	C_d	C_d % difference to F^*
OPTIMA2D	coarse	0.2225	2.4	0.01324	43.2
	medium	0.2266	0.61	0.00997	7.9
	fine	0.2277	0.13	0.00938	1.4
	order, p	2.00	-	2.45	-
	F^*	0.2280	-	0.00924	-
DIABLO	coarse	0.2213	2.6	0.01249	35.9
	medium	0.2255	0.79	0.00965	5.0
	fine	0.2268	0.22	0.00926	0.7
	order, p	1.76	-	2.84	-
	F^*	0.2273	-	0.00919	-

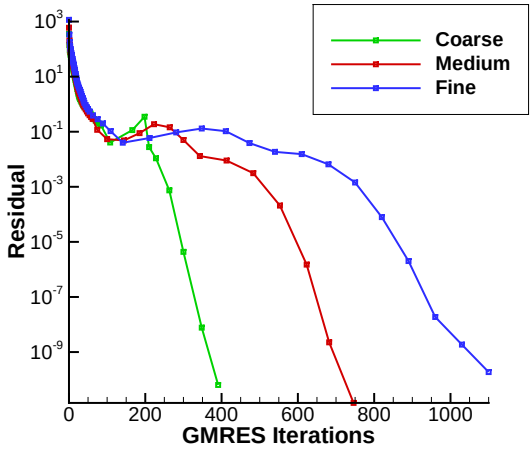


Figure 2: Convergence history for three grid levels

Table 3: Flow conditions used for RAE 2822 airfoil cases

Case	Mach number	Reynolds number	Angle of attack, degrees
1	0.30	7.48e6	0 to 6
2	0.50	12.46e6	0 to 6
3	0.70	17.45e6	0 to 6
4	0.80	19.94e6	0 to 6
5	0.20	18.46e6	0 to 17

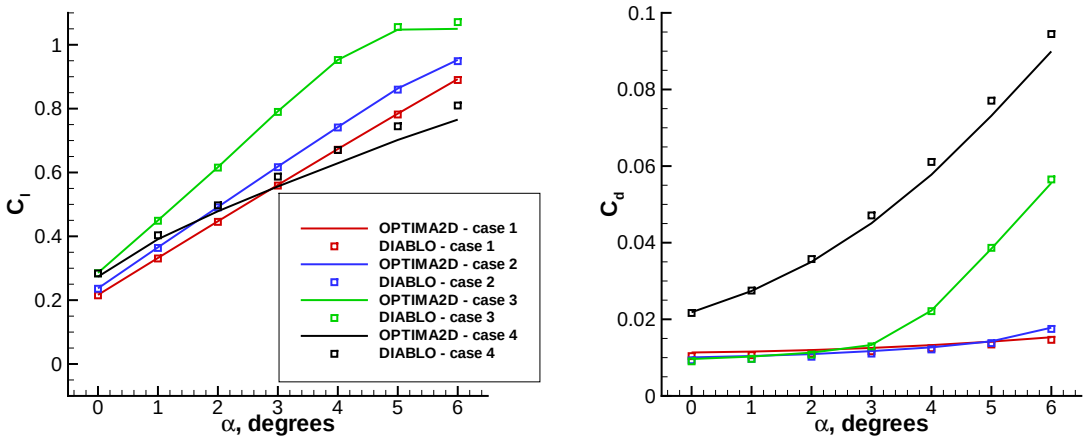
Figure 2 presents convergence histories for all three grid levels in DIABLO. The plot shows the number of linear (GMRES) iterations in both solution phases. The symbols along the lines represent the outer iterations. As can be seen, the solver performs well on all three refinement levels. In particular, the figure highlights the two phases the flow solver employs, with the initial approximate-Newton phase taking many outer iterations to drop the residual 3 orders of magnitude, after which the inexact-Newton phase takes only a few outer iterations to fully converge the solution.

To assess the robustness of DIABLO, flow solutions were performed at various flow conditions around the RAE 2822 airfoil. The grid consists of 289×65 nodes, with an off-wall spacing of 2e-6 chord units. The flow conditions are summarized in Table 3. Converged solutions were obtained with OPTIMA2D as well as the current algorithm. A comparison of the coefficients of lift and drag, C_l and C_d, obtained by both solvers for cases 1 through 4 is presented in Figure 3, showing very good correspondence. Again, any discrepancy can be attributed to the minor differences in the spatial discretizations used in the two solvers and the use of the thin-layer approximation in OPTIMA2D. This comparison demonstrates the applicability of the current algorithm over a range of flow conditions, spanning the subsonic and transonic flow regimes. Figure 4 presents the C_l and C_d comparison for case 5, again showing good correspondence between DIABLO and OPTIMA2D. The largest discrepancy can be seen near the area of maximum lift, where the thin-layer approximation made in OPTIMA2D becomes less accurate due to flow separation.

B. Three-Dimensional Solutions

A three-dimensional flow solution was performed, comparing the results obtained by the current algorithm, DIABLO, against the well-known experimental data of Schmitt and Charpin.[40] The flow conditions used in the experiment were

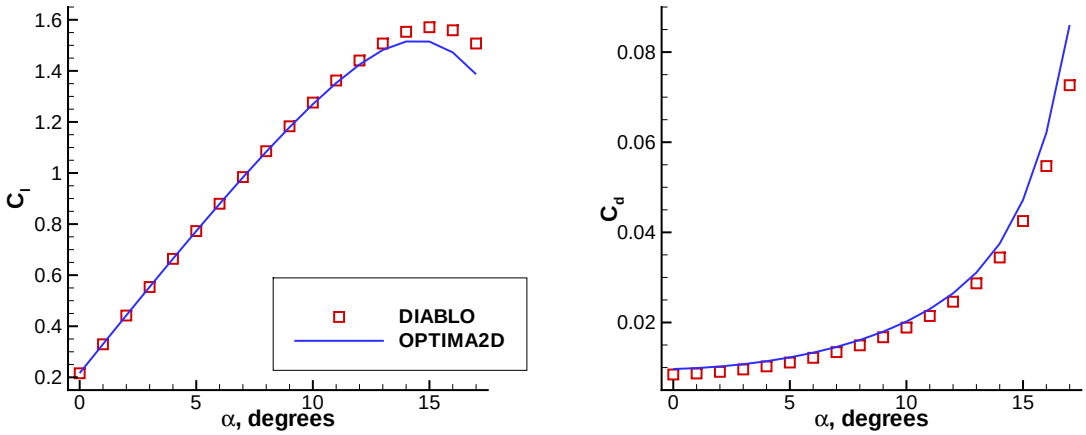
$$\text{Ma} = 0.8395, \text{Re} = 11.72\text{e}6, \alpha = 3.06^\circ.$$



(a) Lift values

(b) Drag values

Figure 3: Lift and drag coefficient comparison for RAE 2822 flow solutions

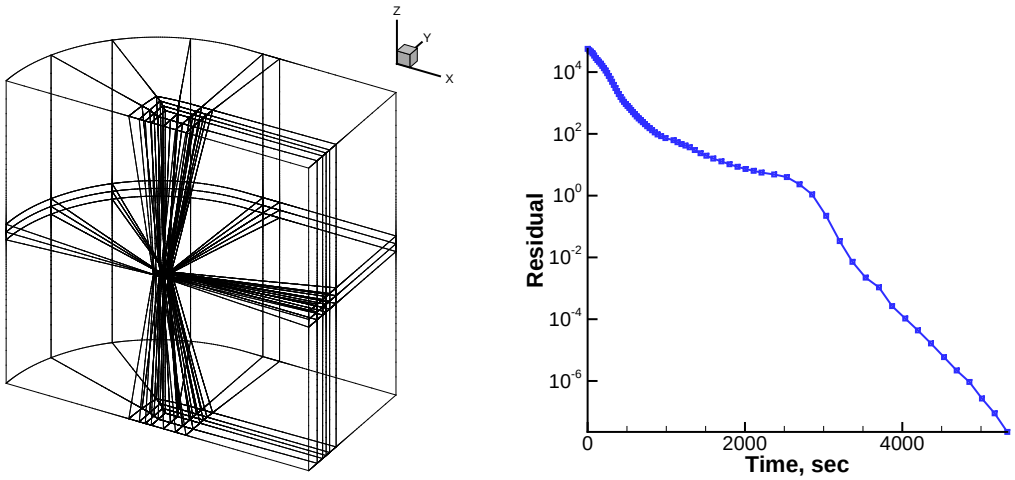


(a) Lift values

(b) Drag values

Figure 4: Lift and drag coefficient comparison for M = 0.2, Re = 18.46e6

The computation was performed on a 128-block C-H topology grid, with a total of 15.1 million nodes and an off-wall spacing of 9e-7 chord units. Each block consists of $49 \times 49 \times 49$ nodes, resulting in good load balancing for parallel computations. The grid layout can be seen in Figure 5a, while Figure 5b presents the convergence history. The residual is reduced by 12 orders of magnitude in 89 minutes when the flow solution is computed on 128 processors. The convergence plot displays the progress of the two phases of the flow solver. The initial approximate-Newton phase slowly decreases the residual by 4 orders of magnitude. This happens at roughly the 2200 second mark, at which point the inexact-Newton approach is used to solve the system fully, dropping the residual a further 8 orders of magnitude in nineteen outer iterations. Figure 6 shows the two shocks that develop on the top surface of the wing, together forming a λ-shock.



(a) Grid layout

(b) Convergence history

Figure 5: Grid and convergence history for transonic 3D flow around ONERA M6 wing

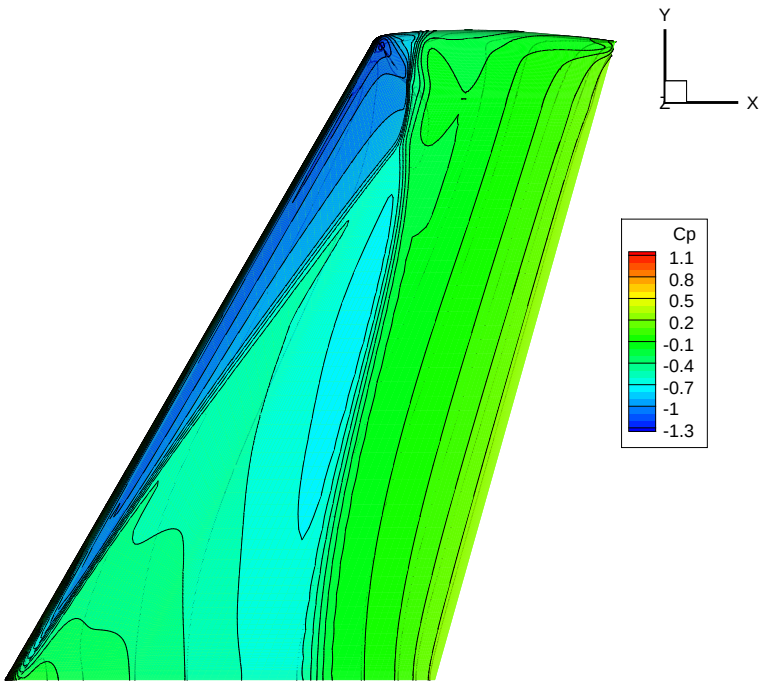
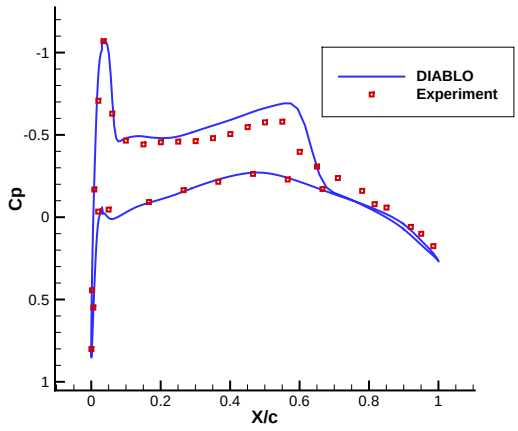
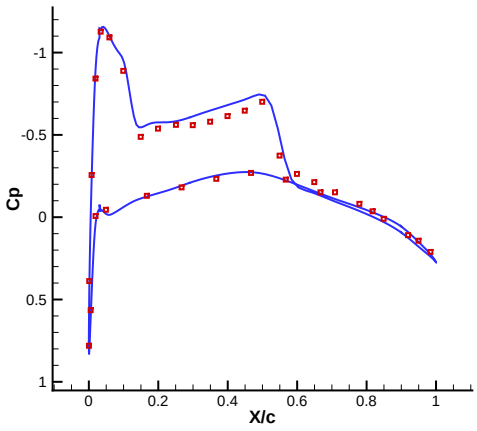


Figure 6: C_p contours on top surface of the ONERA M6 wing

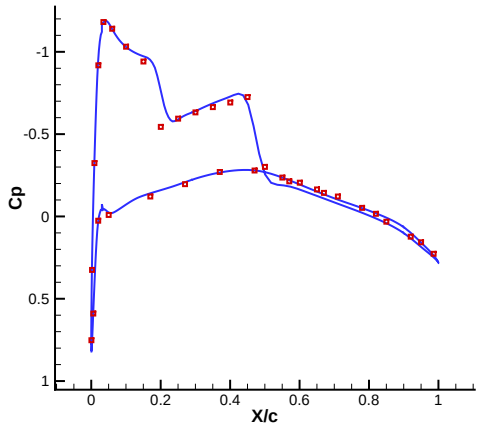
In order to ascertain the accuracy with which DIABLO can predict the transonic flow around the ONERA M6 wing, a comparison to the experimental coefficients of pressure, C_p, is presented in Figure 7. This figure shows the C_p contours at six span-wise sections of the wing, comparing the experimental and computational results. As can be seen, we get very close correspondence between the two sets of data, with the flow solver capturing both shocks on the upper surface of the wing. On this grid, the solution is not fully grid converged.



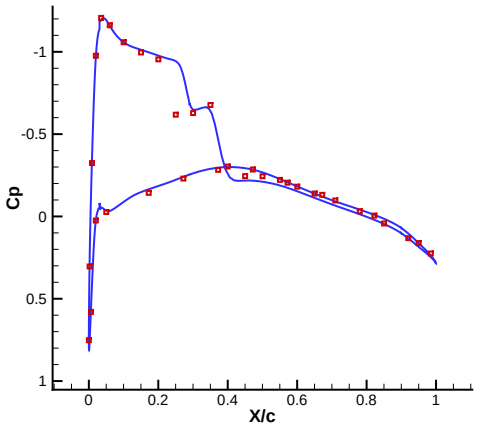
20% span



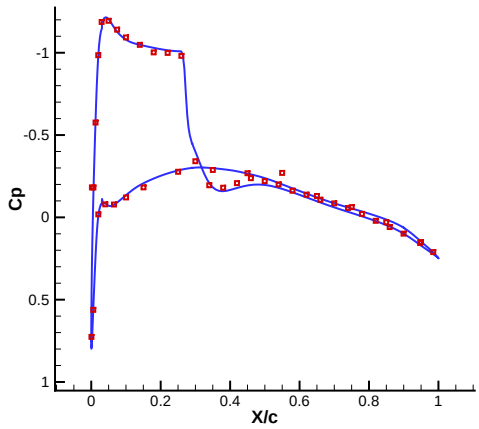
44% span



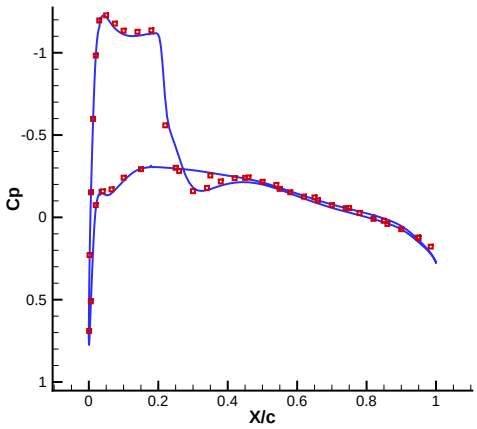
65% span



80% span

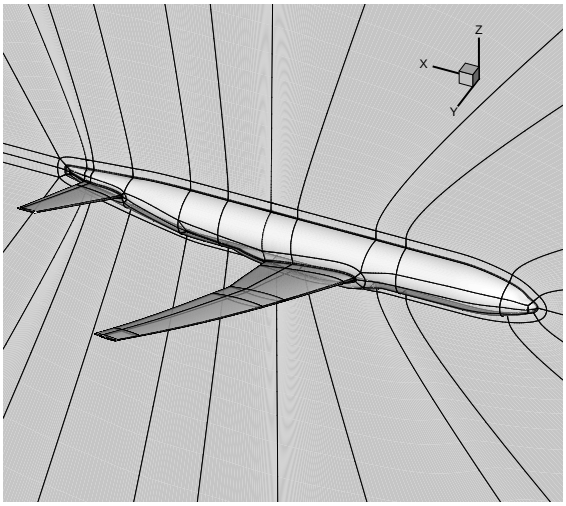


90% span

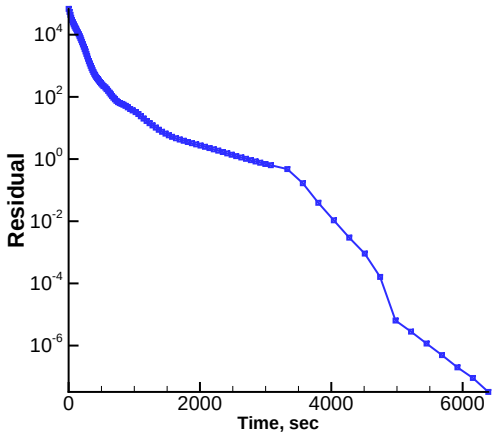


95% span

Figure 7: Experimental and computational C_p distributions for transonic ONERA M6 flow



(a) CRM geometry blocking



(b) Convergence history

Figure 8: Blocking and convergence history for transonic 3D flow around CRM configuration

The CRM flow solution was performed on a computational mesh consisting of 569 blocks, with a total of 10.1 million nodes and an off-wall spacing of 1.8e-6 mean aerodynamic chord units. Due to the complexities of generating a grid around a wing-body configuration, the blocks in this grid are of varying size. The largest block contains $44 \times 41 \times 43$ nodes, while the smallest blocks contain $11 \times 11 \times 15$ nodes. The surface and symmetry plane blocking can be seen in Figure 8a. The flow conditions are

$$\text{Ma} = 0.85, \text{Re} = 5.00\text{e}6, \text{C}_\text{L} = 0.5,$$

which, on this grid, is achieved with an angle of attack of 2.356°.

Figure 8b presents the convergence history for this computation, showing that the residual was reduced by 12 orders of magnitude in approximately 107 minutes using 569 processors. Again, the two convergence phases are clearly visible, with the symbols on the plot representing the individual outer iterations. It should be noted that for this more difficult case, the residual needs to be reduced slightly further (by an initial 5 orders of magnitude) before the switch to the inexact-Newton phase can take place. Otherwise, divergence occurs in the inexact-Newton phase.

Finally, Figure 9 presents the C_p values along the top surface of the CRM configuration. A shock can be seen on the top surface of the main wing, approximately at the mid-chord. The solution compares well with other numerical solutions, such as those found in references 41, 42, and 43.

C. Parallel Performance

In order to characterize the parallel performance of the current algorithm, a 8192-block C-H topology grid was created around the ONERA M6 wing, with a total of 40.2 million nodes and an off-wall spacing of 8e-7. Each block in the computational domain consists of $17 \times 17 \times 17$ nodes, providing ideal load balancing for parallel computations. For flow solutions where fewer processors than computational blocks are used, the blocks are divided evenly among the available processors. All runs were performed on the general-purpose cluster of SciNet, which uses Nehalem processors interconnected with non-blocking 4x-DDR InfiniBand. The flow conditions are

$$\text{Ma} = 0.30, \text{Re} = 7.48\text{e}6, \alpha = 2.0°.$$

The results presented in Figure 10 show that the algorithm scales well with the number of processors. The relative efficiency comparison, based on the amount of time required to obtain the solution with 128 processors, demonstrates that the algorithm performs very well even as more processors are added to the computation. Additionally, the plots also show the excellent parallel performance of the approximate-Schur preconditioner, as demonstrated by the approximately constant number of linear iterations required to obtain

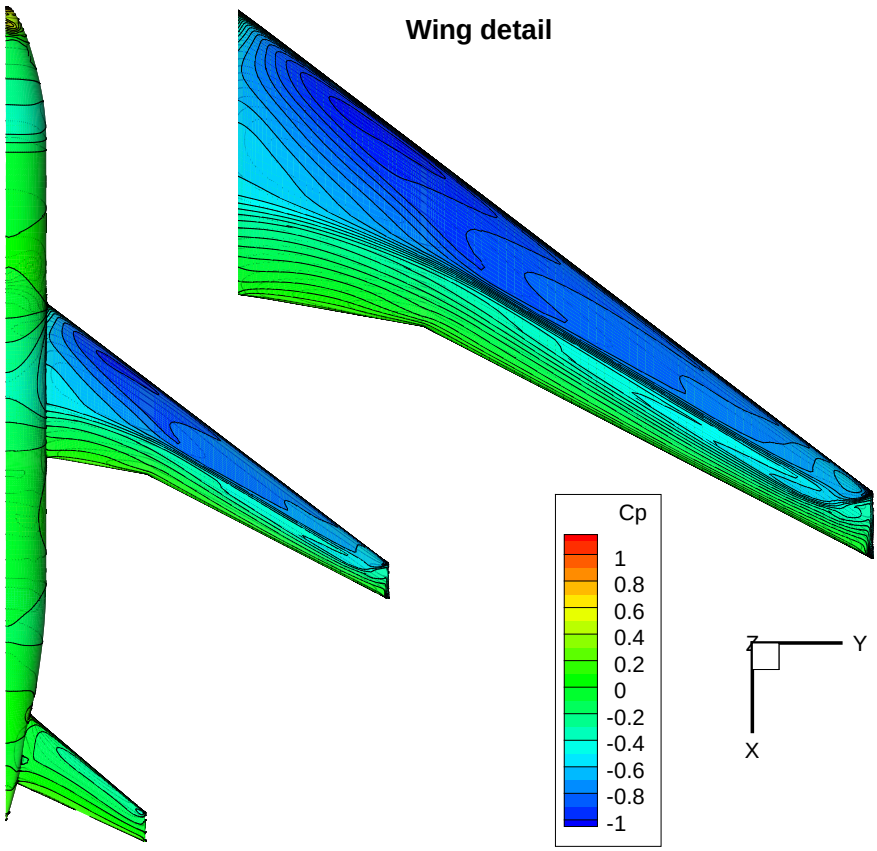


Figure 9: C_p contours on top surface of CRM configuration

a converged flow solution on the various numbers of processors used. Using 4096 processors, the algorithm reduces the residual by 12 orders of magnitude in 23 minutes, while convergence to 3 significant figures in force coefficients is achieved in 8 minutes.

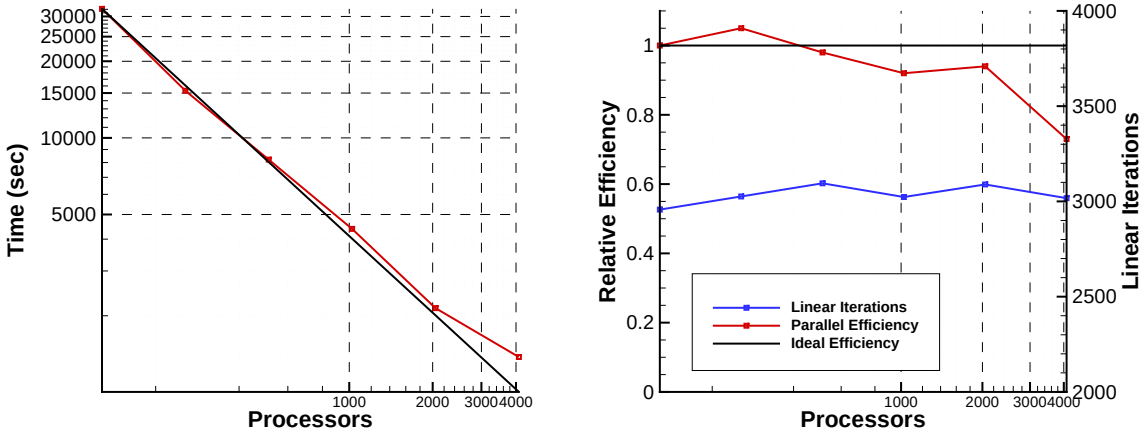
VI. Conclusions

The parallel Newton-Krylov-Schur algorithm with the SBP-SAT discretization was shown to provide accurate and efficient flow solutions to the three-dimensional Navier-Stokes equations and the one-equation Spalart-Allmaras turbulence model. The SAT method of enforcing boundary conditions and inter-block connectivity was successfully applied to the turbulence model, allowing the multi-block solution algorithm to be applied to fully turbulent flows. Special scaling techniques are applied in order to obtain fast convergence when the turbulence model equation is included.

The solutions of two-dimensional flows around the NACA 0012 and RAE 2822 airfoils demonstrate the robustness of the algorithm in obtaining flow solutions over a wide range of flow conditions. Three-dimensional transonic flow solutions around the ONERA M6 wing and the Common Research Model wing-body-tail configuration highlight the ability of the solver to calculate complex three-dimensional flows on multi-block structured grids efficiently. The algorithm shows good parallel efficiency in the range of processors considered (up to 4096).

Comparison to well-known experimental and numerical results for three-dimensional transonic flows demonstrates the ability of the solver to capture complex three-dimensional flow physics, including multiple shocks on the wing surface, accurately.

Future work will involve the implementation of a higher-order SBP spatial discretization.



(a) Time for converged solutions

(b) Relative efficiency and linear iterations

Figure 10: Parallel scaling performance

Acknowledgments

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

Computations were performed on the GPC supercomputer at the SciNet HPC Consortium. SciNet is funded by: the Canada Foundation for Innovation under the auspices of Compute Canada; the Government of Ontario; Ontario Research Fund - Research Excellence; and the University of Toronto.

Appendix

The following presents the complete coordinate transformation needed for the Spalart-Allmaras one-equation turbulence model, which is given by (4). The coordinate transformation has to be performed on all terms in the turbulence model that include spatial derivatives. This includes the advective and diffusive terms, as well as the vorticity value, S.

Advective terms

The advective terms, in Cartesian coordinates, are

$$u_i \frac{\partial \tilde{\nu}}{\partial x_i} = u\frac{\partial \tilde{\nu}}{\partial x} + v\frac{\partial \tilde{\nu}}{\partial y} + w\frac{\partial \tilde{\nu}}{\partial z},$$

where (u, v, w) are the three velocity components.

Performing the coordinate transformation $(x, y, z) \rightarrow (\xi, \eta, \zeta)$, the advective terms become

$$\begin{aligned} U_i \frac{\partial \tilde{\nu}}{\partial \xi_i} &= (u\xi_x + v\xi_y + w\xi_z)\frac{\partial \tilde{\nu}}{\partial \xi} + (u\eta_x + v\eta_y + w\eta_z)\frac{\partial \tilde{\nu}}{\partial \eta} + (u\zeta_x + v\zeta_y + w\zeta_z)\frac{\partial \tilde{\nu}}{\partial \zeta} \\ &= U\frac{\partial \tilde{\nu}}{\partial \xi} + V\frac{\partial \tilde{\nu}}{\partial \eta} + W\frac{\partial \tilde{\nu}}{\partial \zeta}, \end{aligned}$$

where (U, V, W) are the contravariant velocities.

Diffusive terms

Due to the relative complexity of the diffusive terms, only the final results of the coordinate transformation will be presented. It should be noted, however, that the cross-derivative terms are neglected.

The model shown above contains two separate diffusive terms, each of which possesses a slightly different form. The first term is transformed into curvilinear coordinates as follows (neglecting cross-derivatives)

$$\begin{aligned}\nabla\cdot[(\nu+\tilde{\nu})\nabla\tilde{\nu}] &= \partial_x\left[(\nu+\tilde{\nu})\partial_x\tilde{\nu}\right]+\partial_y\left[(\nu+\tilde{\nu})\partial_y\tilde{\nu}\right]+\partial_z\left[(\nu+\tilde{\nu})\partial_z\tilde{\nu}\right]\\ &= \xi_x\partial_\xi[(\nu+\tilde{\nu})\xi_x\partial_\xi\tilde{\nu}]+\xi_y\partial_\xi[(\nu+\tilde{\nu})\xi_y\partial_\xi\tilde{\nu}]+\xi_z\partial_\xi[(\nu+\tilde{\nu})\xi_z\partial_\xi\tilde{\nu}]+\\ &\quad \eta_x\partial_\eta[(\nu+\tilde{\nu})\eta_x\partial_\eta\tilde{\nu}]+\eta_y\partial_\eta[(\nu+\tilde{\nu})\eta_y\partial_\eta\tilde{\nu}]+\eta_z\partial_\eta[(\nu+\tilde{\nu})\eta_z\partial_\eta\tilde{\nu}]+\\ &\quad \zeta_x\partial_\zeta[(\nu+\tilde{\nu})\zeta_x\partial_\zeta\tilde{\nu}]+\zeta_y\partial_\zeta[(\nu+\tilde{\nu})\zeta_y\partial_\zeta\tilde{\nu}]+\zeta_z\partial_\zeta[(\nu+\tilde{\nu})\zeta_z\partial_\zeta\tilde{\nu}].\end{aligned}$$

Similarly, the second term can be transformed into the following form (neglecting cross-derivatives)

$$\begin{aligned}\nabla^2\tilde{\nu} &= \frac{\partial^2\tilde{\nu}}{\partial x^2}+\frac{\partial^2\tilde{\nu}}{\partial y^2}+\frac{\partial^2\tilde{\nu}}{\partial z^2}\\ &= \xi_x\partial_\xi[\xi_x\partial_\xi\tilde{\nu}]+\xi_y\partial_\xi[\xi_y\partial_\xi\tilde{\nu}]+\xi_z\partial_\xi[\xi_z\partial_\xi\tilde{\nu}]+\\ &\quad \eta_x\partial_\eta[\eta_x\partial_\eta\tilde{\nu}]+\eta_y\partial_\eta[\eta_y\partial_\eta\tilde{\nu}]+\eta_z\partial_\eta[\eta_z\partial_\eta\tilde{\nu}]+\\ &\quad \zeta_x\partial_\zeta[\zeta_x\partial_\zeta\tilde{\nu}]+\zeta_y\partial_\zeta[\zeta_y\partial_\zeta\tilde{\nu}]+\zeta_z\partial_\zeta[\zeta_z\partial_\zeta\tilde{\nu}].\end{aligned}$$

Vorticity

The vorticity, S, which contains spatial derivatives of the three velocity components, can be expressed in Cartesian coordinates as

$$S=\left[\left(\frac{\partial w}{\partial y}-\frac{\partial v}{\partial z}\right)^2+\left(\frac{\partial u}{\partial z}-\frac{\partial w}{\partial x}\right)^2+\left(\frac{\partial v}{\partial x}-\frac{\partial u}{\partial y}\right)^2\right]^{-\frac{1}{2}}.$$

After performing the coordinate transformation, the vorticity becomes

$$S=\left(S_1^2+S_2^2+S_3^2\right)^{-\frac{1}{2}},$$

where

$$\begin{aligned}S_1&=(\xi_y w_\xi+\eta_y w_\eta+\zeta_y w_\zeta)-(\xi_z v_\xi+\eta_z v_\eta+\zeta_z v_\zeta),\\ S_2&=(\xi_z u_\xi+\eta_z u_\eta+\zeta_z u_\zeta)-(\xi_x w_\xi+\eta_x w_\eta+\zeta_x w_\zeta),\\ S_3&=(\xi_x v_\xi+\eta_x v_\eta+\zeta_x v_\zeta)-(\xi_y u_\xi+\eta_y u_\eta+\zeta_y u_\zeta).\end{aligned}$$

References

[1]Jespersen, D., Pulliam, T., and Buning, P., "Recent enhacement to OVERFLOW (Navier-Stokes code)," *35th AIAA Aerospace Sciences Meeting and Exhibit*, No. AIAA–97–0644, Reno, Nevada, Jan. 1997.

[2]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*, San Diego, California, United States, 1995, AIAA–95–1733.

[3]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*, No. AIAA–2005–0318, Reno, Nevada, Jan. 2005.

[4]Mavriplis, D. J., "Grid Resolution of a Drag Prediction Workshop using the NSU3D Unstructured Mesh Solver," *17th AIAA Computational Fluid Dynamics Conference*, No. AIAA–2005–4729, Toronto, Canada, June 2005.

[5]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.

[6]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*, No. AIAA–2009–3657, San Antonio, Texas, United States, June 2009.

[7]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*, No. AIAA–2010–116, Orlando, Florida, United States, Jan. 2010.

[8]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.

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

[10]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.

[11]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.

[12]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.

[13]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.

[14]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.

[15]Eliasson, P., Eriksson, S., and Nordström, J., "The Influence of Weak and Strong Solid Wall Boundary Conditions on the Convergence to Steady-State of the Navier-Stokes Equations," *19th AIAA Computational Fluid Dynamics Conference*, No. AIAA–2009–3551, San Antonio, Texas, United States, June 2009.

[16]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.

[17]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, 2010.

[18]Spalart, P. R. and Allmaras, S. R., "A One-Equation Turbulence Model for Aerodynamic Flows," *30th AIAA Aerospace Sciences Meeting and Exhibit*, No. AIAA–92–0439, Reno, Nevada, United States, Jan. 1992.

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

[20]Osusky, M. and Zingg, D. W., "A parallel multi-block Newton-Krylov flow solver for the Navier-Stokes equations with SBP-SAT discretization," *18th Annual Conference of the CFD Society of Canada*, London, Ontario, Canada, May 2010.

[21]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.

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

[23]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*, No. AIAA–81–1259, Palo Alto, California, United States, 1981.

[24]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.

[25]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.

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

[27]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*, No. AIAA–2005–5230, Toronto, Canada, June 2005.

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

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

[30]Kam, D. C. W., *A three-dimensional Newton-Krylov Navier-Stokes flow solver using a one-equation turbulence model*, Master's thesis, University of Toronto, Toronto, Ontario, Canada, 2007.

[31]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*, No. AIAA–93–0878, Reno, Nevada, 1993.

[32]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.

[33]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.

[34]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.

[35]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.

[36]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.

[37]Zingg, D. W., De Rango, S., Nemec, M., and Pulliam, T. H., "Comparison of several spatial discretizations for the Navier-Stokes equations," *Journal of Computational Physics*, Vol. 160, No. 2, May 2000, pp. 683–704.

[38]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*, No. AIAA–2008–6919, Honolulu, Hawaii, United States, Aug. 2008.

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

[40]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.

[41]Vassberg, J. C., Tinoco, E. N., Mani, M., Rider, B., Zickuhr, T., Levy, D. W., Brodersen, O. P., Eisfeld, B., Crippa, S., Wahls, R. A., Morrison, J. H., Mavriplis, D. J., and Murayama, M., "Summary of the Fourth AIAA CFD Drag Prediction Workshop," *28th AIAA Applied Aerodynamics Conference*, No. AIAA–2010–4547, Chicago, Illinois, United States, June 2010.

[42]Mavriplis, D. and Long, M., "NSU3D Results for the Fourth AIAA Drag Prediction Workshop," *28th AIAA Applied Aerodynamics Conference*, No. AIAA–2010–4550, Chicago, Illinois, United States, June 2010.

[43]Lee-Rausch, E. M., Hammond, D. P., Nielsen, E. J., Pirzadeh, S. Z., and Rumsey, C. L., "Application of the FUN3D Unstructured-Grid Navier-Stokes Solver to the 4th AIAA Drag Prediction Workshop Cases," *28th AIAA Applied Aerodynamics Conference*, No. AIAA–2010–4551, Chicago, Illinois, United States, June 2010.